Mälardalen University Press Dissertations
No. 133

# NETWORK OVERLOAD AVOIDANCE BY TRAFFIC ENGINEERING AND CONTENT CACHING

**Henrik Abrahamsson**

**2012**


School of Innovation, Design and Engineering

ISBN 978-91-7585-087-1
ISSN 1651-4238
Printed by Mälardalen University, Västerås, Sweden

Swedish Institute of Computer Science
Doctoral Thesis
SICS Dissertation Series 58

# Network Overload Avoidance by Traffic Engineering and Content Caching

Henrik Abrahamsson

2012

Swedish Institute of Computer Science
Stockholm, Sweden

ISSN 1101-1335
ISRN SICS-D–58–SE
Printed by Mälardalen University, Västerås, Sweden

Mälardalen University Press Dissertations
No. 133

# NETWORK OVERLOAD AVOIDANCE BY TRAFFIC ENGINEERING AND CONTENT CACHING

Henrik Abrahamsson

Akademisk avhandling

som för avläggande av teknologie doktorsexamen i datavetenskap vid
Akademin för innovation, design och teknik kommer att offentligen försvaras
onsdagen den 19 december 2012, 13.15 i Kappa, Mälardalens högskola, Västerås.

Fakultetsopponent: Doctor Luca Muscariello, Orange Telecom


Akademin för innovation, design och teknik

Abstract

The Internet traffic volume continues to grow at a great rate, now driven by video and TV distribution. For network operators it is important to avoid congestion in the network, and to meet service level agreements with their customers. This thesis presents work on two methods operators can use to reduce links loads in their networks: traffic engineering and content caching.

This thesis studies access patterns for TV and video and the potential for caching. The investigation is done both using simulation and by analysis of logs from a large TV-on-Demand system over four months.

The results show that there is a small set of programs that account for a large fraction of the requests and that a comparatively small local cache can be used to significantly reduce the peak link loads during prime time. The investigation also demonstrates how the popularity of programs changes over time and shows that the access pattern in a TV-on-Demand system very much depends on the content type.

For traffic engineering the objective is to avoid congestion in the network and to make better use of available resources by adapting the routing to the current traffic situation. The main challenge for traffic engineering in IP networks is to cope with the dynamics of Internet traffic demands.

This thesis proposes L-balanced routings that route the traffic on the shortest paths possible but make sure that no link is utilised to more than a given level L. L-balanced routing gives efficient routing of traffic and controlled spare capacity to handle unpredictable changes in traffic. We present an L-balanced routing algorithm and a heuristic search method for finding L-balanced weight settings for the legacy routing protocols OSPF and IS-IS. We show that the search and the resulting weight settings work well in real network scenarios.

ISBN 978-91-7585-087-1
ISSN 1651-4238

# Abstract

The Internet traffic volume continues to grow at a great rate, now driven by video and TV distribution. For network operators it is important to avoid congestion in the network, and to meet service level agreements with their customers. This thesis presents work on two methods operators can use to reduce links loads in their networks: traffic engineering and content caching.

This thesis studies access patterns for TV and video and the potential for caching. The investigation is done both using simulation and by analysis of logs from a large TV-on-Demand system over four months.

The results show that there is a small set of programs that account for a large fraction of the requests and that a comparatively small local cache can be used to significantly reduce the peak link loads during prime time. The investigation also demonstrates how the popularity of programs changes over time and shows that the access pattern in a TV-on-Demand system very much depends on the content type.

For traffic engineering the objective is to avoid congestion in the network and to make better use of available resources by adapting the routing to the current traffic situation. The main challenge for traffic engineering in IP networks is to cope with the dynamics of Internet traffic demands.

This thesis proposes L-balanced routings that route the traffic on the shortest paths possible but make sure that no link is utilised to more than a given level L. L-balanced routing gives efficient routing of traffic and controlled spare capacity to handle unpredictable changes in traffic. We present an L-balanced routing algorithm and a heuristic search method for finding L-balanced weight settings for the legacy routing protocols OSPF and IS-IS. We show that the search and the resulting weight settings work well in real network scenarios.

# Sammanfattning

Trafiken på Internet fortsätter att växa i snabb takt nu pådrivet av TV- och videodistribution över nätet. För nätverksoperatörer är det viktigt att förstå och hantera trafikbeteendet för att undvika överlast i nätet och för att kunna tillhandahålla kommunikationstjänster av god kvalitet. Den här avhandlingen handlar om två olika tillvägagångssätt för att undvika överlast i nätet: lastbalansering och lokal mellanlagring.

I den här avhandlingen undersöks användarbeteenden och efterfrågemönster för TV och video och potentialen för lokal mellanlagring. Undersökningen görs dels med simulering och dels genom analys av loggar från ett stort TV-system.

Resultaten visar att det är en liten andel av programmen som står för en stor del av efterfrågan. I många fall kan man hantera 50% av efterfrågan genom att lagra 5% av utbudet. Studien visar också att programutbud och genre har stor inverkan på efterfrågemönster och på hur snabbt programmen avtar i popularitet. Det är också stora dygnsvariationer i efterfrågan och det är viktigt att lagra rätt program för att hantera toppar i efterfrågan under kvällstid.

För lastbalansering i IP-nätverk är målet att kunna anpassa vägvalet efter den aktuella trafiksituationen och balansera trafiken över flera vägar genom nätverket om det behövs. Man kan på så vis utnyttja nätverket mer effektivt och undvika överlast. Utmaningen ligger i att Internettrafik ofta är skurig med stora variationer i trafikens mängd och riktning.

I den här avhandlingen föreslås så kallat L-balanserat vägval där trafiken skickas kortast möjliga väg men man ser till att ingen länk lastas till mer än en given nivå L. L-balanserat vägval ger en kontrollerad reservkapacitet för att hantera oförutsägbara förändringar i trafiken. Vi presenterar en L-balanserad vägvalsalgoritm samt en heuristisk sökmetod för att hitta L-balanserade viktsättningar i vägvalsprotokollen OSPF och IS-IS. Vi visar att sökmetoden och de resulterande viktinställningarna fungerar väl i verkliga nätverkscenarier.

# Acknowledgements

During the years that I have worked on this thesis, I have been involved in many different projects and collaborations. I have on numerous occasions discussed the research with colleagues both in Kista and Västerås, but also at project meetings, workshops and conferences in many other places: from Nanjing and Beijing to Ottawa, Boston, Madrid and Zurich, to name a few. I want to thank all research colleagues who have been involved in the discussions that have helped to shape the ideas that I present in this thesis.

I especially want to thank my advisor Mats Björkman. Without his encouragement, guidance and support this thesis would never have been finished. I also want to thank my co-advisor Bengt Ahlgren for his support and for giving me the opportunity to work in the NETS group and to do research at SICS.

I want to thank Anders Gunnar and Per Kreuger for many interesting discussions on all aspects of traffic and cache management and for great collaboration on many research papers and project deliverables. I would also like to thank Adam Dunkels for many inspiring discussions over the years.

Thanks also to Mattias Nordmark and Göran Olofsson for our collaboration on access patterns for TV-on-Demand, which also made it possible to write the final paper and complete this thesis.

Many thanks also to Ian Marsh, Laura Feeney, Anders Lindgren, Mudassar Aslam, Oliver Schwarz, and all other, current and former, colleagues at SICS for your support and for creating a stimulating work environment.

*Much of the work in this thesis has been performed within the SICS Center for Networked Systems funded by VINNOVA, KKS, SSF, ABB, Ericsson, Saab SDS, TeliaSonera, T2Data, Vendolocus and Peerialism.*

# List of publications

**Publications included in the thesis**

1. Henrik Abrahamsson, Juan Alonso, Bengt Ahlgren, Anders Andersson and Per Kreuger. A Multi Path Routing Algorithm for IP Networks Based on Flow Optimisation. In *Proceedings of Third COST 263 International Workshop on Quality of Future Internet Services(QoFIS 2002)*, Zurich, Switzerland, October 2002.

2. Henrik Abrahamsson and Mats Björkman. Robust Traffic Engineering using L-balanced Weight-Settings in OSPF/ISIS. In: *Sixth International Conference on Broadband Communications, Networks, and Systems (BROADNETS 2009)*, Madrid, Spain, September 2009.

3. Henrik Abrahamsson and Mats Björkman. Simulation of IPTV caching strategies. In: *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'10)*, Ottawa, Canada, July 2010.

4. Henrik Abrahamsson and Mats Björkman. Caching for IPTV distribution with time-shift. In: *International conference on Computing, Networking & Communications (ICNC'13)*, January 2013, San Diego, USA.

5. Henrik Abrahamsson and Mattias Nordmark. Program popularity and viewer behaviour in a large TV-on-Demand system. In: *Internet Measurement Conference (IMC'12)*, Boston, USA, November 2012.

**Other publications**

In addition to the papers included in the thesis I have also co-authored the following papers:

1. Simon Schütz, Henrik Abrahamsson, Bengt Ahlgren, and Marcus Brunner, Design and Implementation of the Node Identity Internetworking Architecture. In *Computer Networks*, 54(7):1142-1154, May, 2010.

2. Per Kreuger and Henrik Abrahamsson, Scheduling IPTV content predistribution, In *Proceedings of the 9th IEEE International Workshop on IP Operations and Management (IPOM'09)*, Venice, Italy, October, 2009.

3. Henrik Abrahamsson and Per Kreuger, A Case for Resource Management in IPTV distribution, In *Proceedings of the 5th Swedish National Computer Networking Workshop (SNCNW)*, Karlskrona, Sweden, April 2008.

4. G. Leduc, H. Abrahamsson, S. Balon, S. Bessler, M. D'Arienzo, O. Delcourt, J. Domingo-Pascual, S. Cerav-Erbas, I. Gojmerac, X. Masip, A. Pescapè, B. Quoitin, S.F. Romano, E. Salvatori, F. Skivée, H.T. Tran, S. Uhlig, and H. Ümit. An open source traffic engineering toolbox , In *Computer Communications*, 29(5):593-610, March, 2006.

5. Anders Gunnar, Henrik Abrahamsson, and Mattias Söderqvist. Performance of Traffic Engineering in Operational IP-Networks - An Experimental Study, In *Proceedings of the 5th IEEE International Workshop on IP Operations and Management (IPOM'05)*, Barcelona, Spain, October, 2005.

6. M. Brunner, A. Galis, L. Cheng, J. Andres Colas, B. Ahlgren, A. Gunnar, H. Abrahamsson, R. Szabo, S. Csaba, J. Nielsen, S. Schütz A. Gonzalez Prieto, R. Stadler, and G. Molnar, Towards Ambient Networks Management, In *Proceedings of the Second International Workshop on Mobility Aware Technologies and Applications (MATA 2005)*, Montreal, Canada, October, 2005.

7. Henrik Abrahamsson and Anders Gunnar, Traffic Engineering in Ambient Networks: Challenges and Approaches , In *Proceedings of the Second Swedish National Computer Networking Workshop (SNCNW)*, Karlstad, Sweden, November, 2004.

8. M. Brunner, A. Galis, L. Cheng, J. Andres Colas, B. Ahlgren, A. Gunnar, H. Abrahamsson, R. Szabo, S. Csaba, J. Nielsen, A. Gonzalez Prieto, R. Stadler, and G. Molnar, Ambient Networks Management Challenges and Approaches, In *Proceedings of the First International Workshop on Mobility Aware Technologies and Applications (MATA 2004)*, Florianpolis, Brazil, October, 2004.

9. Henrik Abrahamsson, Olof Hagsand, and Ian Marsh. TCP over High Speed Variable Capacity Links: A Simulation Study for Bandwidth Allocation , In *Proceedings of the 7th International Workshop on Protocols For High-Speed Networks (PfHSN 2002)*, Berlin, Germany, April, 2002.

10. Henrik Abrahamsson and Bengt Ahlgren. Using empirical distributions to characterize web client traffic and to generate synthetic traffic. In *Proceedings of IEEE Globecom: Global Internet*, San Francisco, USA, November 2000.

# Contents

# I

# Thesis

# Chapter 1

# Introduction

The Internet is a worldwide communication network that today serves billions of Internet users [1]. It is a giant infrastructure of optical fibres, copper wires and wireless connections that via packet switches connect a wide variety of end-hosts: ranging from servers in data centers to PC:s and laptop computers, to mobile phones and smaller devices embedded in our homes, in cars and in the environment around us. The Internet is also an infrastructure that supports a diversity of applications like the web, mail, file sharing, social networking services, telephony, radio, video and TV distribution, games, banking and commerce of many kinds; and where new applications constantly are developed and deployed.

Internet traffic volumes continue to grow at a great rate. For network operators it is important to avoid congestion in the network, and to meet service level agreements with their customers. This thesis presents work on two methods operators can use to reduce links loads and avoid congestion in their networks: traffic engineering and caching of video and TV content.

## 1.1 Internet – a network of networks

The Internet is a network of networks. It consists of a large number of independently managed networks of different sizes, different capacities, and under different administrations. When you click on a link in your web browser the requested webpage often travels over many different networks, sometimes worldwide, on the way to your computer. The view point in this thesis is often

from one operator network and the challenge of understanding and handling traffic demands to avoid overload in the network.

The structure of the Internet and how traffic flows between networks are changing over time [2], often driven by commercial interests and business agreements. The traditional view is that the networks that constitute the Internet are Internet Service Provider (ISP) networks connected together in a loose hierarchy. At the top there are a small number of tier-1 operators (for instance AT&T, Level 3, and TeliaSonera International carrier [3]) with large international high-capacity networks, that directly connect to each other. The tier-1 operators have peering agreements that allow data to flow between the networks without charging each other for the data transmitted. A tier-2 network is typically a regional or national network. It can have peering agreements with other tier-2 networks to exchange traffic but it is also a customer to one or more tier-1 operators and need to buy transit to reach some parts of the Internet. At the bottom of the network hierarchy are the access networks that connect the end hosts to the Internet. These are typically local telephone companies, university or company networks that in turn are customers to upper-tier networks to be able to communicate worldwide. The hierarchical network structure is also complemented by a very large number of peering connections between networks of different types at Internet exchange points (IXPs) [4, 5]. Networks make peering agreement and exchange traffic based on commercial or other interests, irrespective of network size and tier structure.

In addition to traditional ISP networks, content delivery networks (CDNs) like Akamai and Limelight are well-established since a decade back, and today deliver a large share of the Internet content [6, 7, 8]. More recently, large content providers like Google and Netflix have started to build their own content delivery networks [9, 10, 11, 12, 13].

## 1.2 Traffic characteristics and access patterns

The traffic characteristics in a network depend on when and where on the Internet the traffic is measured. The traffic behaviour in a large backbone network differs from that in a small company network, and the traffic characteristics change with new applications, new types of networks and with changing user behaviour.

The Internet traffic volumes are constantly increasing but both the growth rate and the traffic mix very much depend on where on the Internet the measurements are done. Recent measurements of traffic volumes from large ISPs,

peering routers and Internet exchange points report annual growth rates of 35-100% [2, 5, 14, 15]. Figure 1.1 shows an example of traffic volumes at the Netnod Internet exchange point in Stockholm.

The Internet traffic over the last 15 years has been dominated by web traffic (transfered with the HTTP protocol) and peer-to-peer (P2P) traffic [2, 4, 14, 15, 17, 18, 19, 20, 21]. The share of the traffic volume that is P2P or HTTP traffic differs between different parts of the Internet and has changed over time. Fifteen years ago, measurements on the Internet backbone showed that 70-75% of the traffic was web traffic [22]. After that P2P file sharing applications became popular and contributed to a large share of the traffic volume [14, 15, 17, 20, 21], but many reports from the last couple of years show that HTTP traffic is again increasing. Measurements from large ISPs and peering routers [2, 14], show a decline in the share of P2P traffic and a growth in HTTP to more than 50% of the traffic. Measurements at a large European IXP [4] also show that HTTP accounts for more than 50% of the bytes, but the amount of HTTP traffic varies greatly between different participating AS:es.

Maier et al [18], monitoring 20000 residential DSL customers in 2009, report that HTTP and not P2P dominates the traffic with 57% of the transfered bytes, while other measurements of residential user traffic show that P2P is still dominant but not growing [15, 17].

A large part of the Internet traffic is delivery of video content in different ways: P2P file sharing, P2P streaming services, and much of the increase in web traffic is video that is transfered with HTTP, for instance from sites like Youtube. Video and TV-on-Demand streaming services like Netflix are also becoming increasingly popular. There are reports that Netflix alone represents more than 30% of peak downstream traffic in the US [11, 23].

## 1.3 Television and video over IP

Television and video distribution over IP networks is an area with fast development. There are many terms that describe slightly different aspects of the area: IPTV, Internet television, web TV, TV-on-Demand, time-shifted TV, start-over TV, restart TV, catch-up TV, and so on. Some of these terms can also have different meanings in different contexts.

Internet television is a general term that here means TV programs that are available via the Internet. This includes TV services where traditional TV broadcasters (or others) make TV programs available for on-demand viewing. It also includes live broadcasts of individual programs or entire TV channels

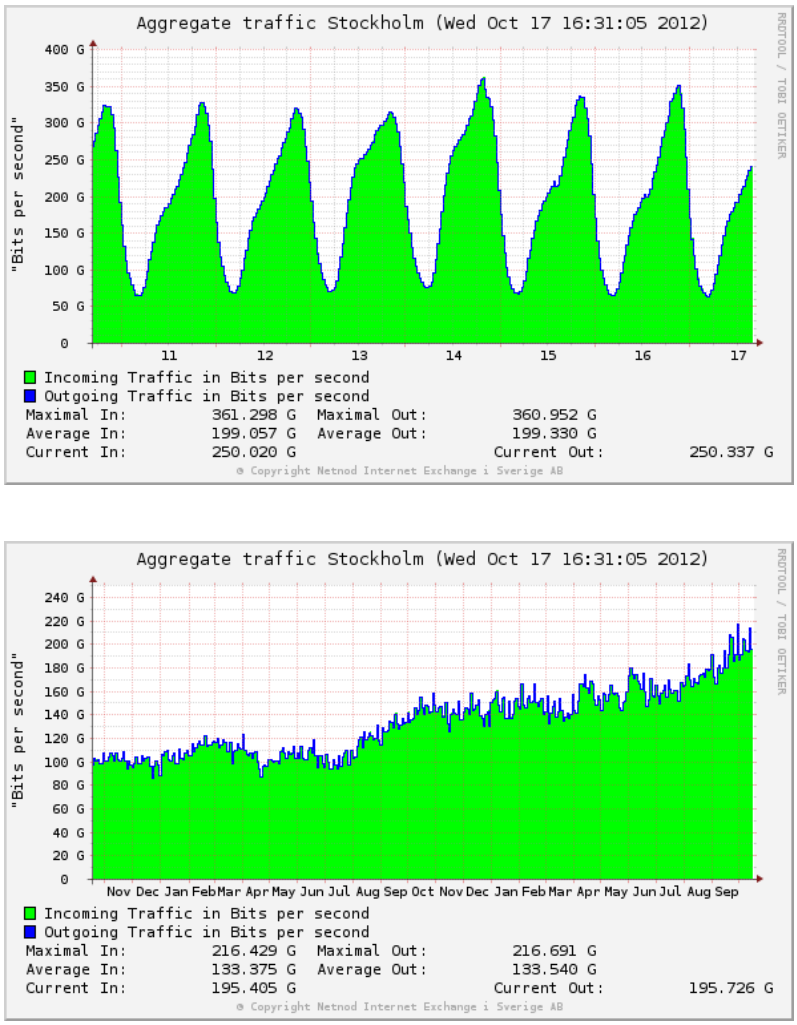



Figure 1.1: Example of Internet traffic at the Netnod Internet exchange point in Stockholm [16] (reprinted with permission). The top graph shows variation over a week (30 minute average) and the bottom graph shows how the traffic volume has increased over two years (one day average).

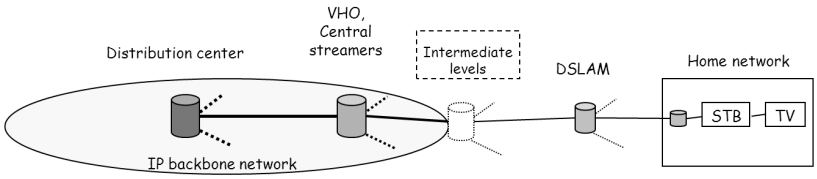



Figure 1.2: IPTV network architecture.

over the Internet.

From an Internet service provider perspective much of the TV and video (for instance from Youtube or Netflix) is so called over-the-top (OTT) content. This means that the operator just delivers the IP packets and does not control the TV and video services.

There are also operator managed services where TV is delivered over an IP network to subscribers. This is usually termed IPTV (Internet Protocol Television). The IPTV service includes traditional TV channels that usually are distributed using IP multicast. The operators often also introduce new on-demand services where viewers can control when to watch the programs. These services differ slightly depending on when the programs become available and for how long they are available. We here use the terms TV-on-Demand and time-shifted TV as general terms for programs that can be viewed decoupled from the traditional TV schedule. Start-over TV and restart TV more specifically means that the viewer can restart and choose to watch an ongoing broadcast program from the beginning. Catch-up TV usually means that programs become available for on-demand viewing some time after the broadcast. An IPTV service often have a mix of these features for different programs depending on agreements with content providers. It is also often combined with a traditional Video-on-Demand service with streaming of rental movies.

When distributing broadcast TV channels using IP multicast there is only one data stream per channel, while for TV-on-Demand there can be one stream per customer. Distributing dedicated TV streams to each viewer requires a lot of bandwidth and server capacity.

One branch of a typical IPTV architecture with a hierarchical tree-like network structure is illustrated in Figure 1.2. The TV content is delivered from content providers and comes into the network at a central distribution center from where it is transmitted to Video Hub Offices (VHO). A Video Hub Office

has storage and video streaming equipment to serve a district or a city. Under the VHO there can be intermediate levels of storage and video servers. Different operators try and use different structures of varying complexity. The figure also shows a TV subscriber with a home network where the TV and the set-top box (STB) is connected via a residential gateway to a Digital Subscriber Line Access Multiplexer (DSLAM). The TV channels are distributed using IP multicast from the distribution center to the set-top boxes. TV programs requested outside the schedule are streamed with unicast from the VHO (or from an intermediate server if available) to the set-top box.

## 1.4 Overload avoidance

### 1.4.1 Traffic management

Internet traffic management means handling the traffic situation in the networks; avoiding congestion and making good use of available network resources.

Traffic management involves both the end hosts and the network operators. It involves the end hosts in that they for many applications run TCP congestion control and adapt the send rate to what the network can handle. TCP increases the send rate to find out the available network capacity. When a packet is lost this is interpreted as network congestion and the transmission rate is decreased. From a network operator perspective traffic management involves monitoring and controlling the traffic behaviour in the network. It also includes traffic engineering where the routing of traffic through the network is adapted to the current traffic situation.

For network operators it is important to manage the traffic situation in the network and meet service level agreements (SLAs) made with their customers. The traffic demands in a network may fluctuate and change over time. Traffic engineering mechanisms can then be used to adapt to the changes in traffic demand and distribute traffic in order to benefit from available network resources.

The first step in the traffic engineering process is to collect the necessary information about network topology and the current traffic situation. Most traffic engineering methods need as input a traffic matrix describing the demand between each pair of nodes in the network. The traffic matrix is then used as input to the routing optimization.

Network operators today have different alternatives for coping with traffic variability: ranging from just over-dimensioning network capacity a lot, to

occasionally tuning the configuration of the routing protocols in order to postpone upgrades of network equipment, to more active use of traffic monitoring and traffic engineering mechanisms to manage the traffic situation.

One of the main alternatives for traffic engineering within an IP network [24, 25] is to use different methods for setting the link costs, and so decide upon the shortest paths, in the routing protocols OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System). These are both link-state protocols where the routing decisions are based on link costs and a shortest (least-cost) path calculation. With the equal-cost multi-path (ECMP) extension to the routing protocols the traffic can also be distributed over several paths that have the same cost. These routing protocols were designed to be simple and robust rather than to optimise the resource usage. They do not by themselves consider network utilisation and do not always make good use of network resources. The traffic is routed on the shortest path through the network even if the shortest path is overloaded and there exist alternative paths. It is up to the operator to find a configuration of the protocol, a set of link costs, that is best suited for the current traffic situation and that avoids congestion in the network.

There are also many other alternatives for how to do traffic engineering. For instance, Multi-Protocol Label Switching (MPLS) [26] has been widely used to control network traffic flows by setting up label-switched paths through the network. More recently, much focus has been on OpenFlow and Software-defined Networking (SDN) with the possibility of fine-grained, flow-based management and control, and the separation of control plane and data plane functionality [27, 28, 29, 30, 31, 32, 33].

### 1.4.2 Caching

One way to reduce the network load is to use caching, where copies of content are stored in local server nodes closer to the clients. By serving requests from the local cache instead of from a central server, repeated transfers of popular content over the network can be avoided.

Caching can be used to reduce network traffic and server load. It can also be used with other objectives: to lower access latency or to increase availability and robustness of a service.

Caching has been widely studied and used for web content [34, 35, 36, 37], for video and TV-on-demand [38, 39, 40, 41, 42] and for content distribution network [6, 7, 8, 9, 10, 11, 12]. Caching, integrated into the network architecture, is also a fundamental component in much of the long term research on future Internet architectures, like Information-centric networking [43, 44].

If we consider caching in a simple hierarchical system, as outlined in Figure 1.2 for IPTV, then a request from a client first goes to the cache, and if the program is not available there it is instead transfered from the central server. The system design parameters include: on what level in the network should the cache be placed, the size of the cache, and what caching policy to use.

The hit ratio, the share of requests that can be served by the cache, depends on the request pattern and on what content is placed in the cache. Given a limited cache size, and content that change in popularity over time, a strategy is needed to decide what should be put in the cache and what should be evicted. Many different cache replacement policies have been proposed in the literature [35, 37]. Two classic eviction policies are Least Recently Used (LRU) and Least Frequently Used (LFU). With the LRU strategy the program that has not been requested for the longest time is deleted from the cache. With LFU the program that is requested least often is discarded.

For the design of a caching system and for the choice of caching strategy, it is important to understand demand and access patterns.

## 1.5 Outline of thesis

This thesis has two parts: an introductory part (Chapters 1 to 5) followed by a collection of five papers. Chapter 2 describes the research issues that this thesis deals with and the scientific contributions of the thesis. Chapter 3 summarizes the papers included in the thesis and their contributions. Chapter 4 discusses related work and put the research into context. In Chapter 5 there are conclusions and future work.

# Chapter 2

# Research Issues and Scientific Contributions

This thesis presents work on traffic engineering and on caching as means to avoid link overload in the network. For traffic engineering the purpose is to develop methods to control and steer the traffic. For caching the idea is to store popular content closer to the users to avoid repeated transfers of identical content. The work is done by simulation and by empirical studies and analysis of access patterns using logs from a real system.

## 2.1 Robust traffic engineering

The objective of traffic engineering is to avoid congestion in the network and to make better use of available resources by adapting the routing to the current traffic situation. The main challenge for traffic engineering is to cope with the dynamics of traffic demands and topology. Traffic is often bursty and there can be unpredictable changes and shifts in traffic demand, for instance due to hotspots and flash crowds, or because a link goes down, there are changes in the inter-domain routing, or because traffic in an overlay is re-directed. For future networks more variability in traffic demands is also expected due to mobility of nodes and networks and more dynamic on-demand service level agreements.

The traffic variability means that, even if we could measure the current traffic situation exactly, it would not always correctly predict the near future traffic situation. Traffic engineering mechanisms need to be robust and able to

handle traffic variability and uncertainties in input traffic data.

### 2.1.1 Contributions

The Papers A and B in this thesis cover different aspects of robust traffic engineering. We propose *l*-balanced routings as a way for an operator to handle traffic variability and uncertainties in input traffic data. An *l*-balanced solution routes the traffic on the shortest paths possible but makes sure that no link is utilised to more than a given level *l*. The contributions are an *l*-balanced routing algorithm based on multi-commodity flow optimisation and a heuristic search method for finding *l*-balanced weight settings for the legacy routing protocols OSPF and IS-IS.

L-balanced routing gives the operator possibility to apply simple rules of thumb for controlling the maximum link utilisation and control the amount of spare capacity needed to handle sudden traffic variations. It gives more controlled traffic levels than other cost functions and more efficient routing for low traffic loads when there is no need to spread traffic over longer paths.

## 2.2 Understanding TV-on-Demand access patterns and their impact on caching

Today video and TV distribution dominate Internet traffic. The increasing demand for high-bandwidth multimedia services put pressure on Internet service providers. It is therefore essential for traffic and cache management to understand TV program popularity and access patterns in real networks.

### 2.2.1 Contributions

The Papers C, D and E in this thesis cover different aspects of TV-on-Demand access patterns and the potential for caching. In Papers C and D we simulate TV distribution with time-shift and investigate what impact TV program popularity, program set size, cache replacement policy and other factors have on the caching efficiency. The simulation results show that introducing a local cache close to the viewers significantly reduces the network load from TV on-demand services. By caching 4% of the program volume we can decrease the peak load during prime time by almost 50%. We also show that the TV program type and how program popularity changes over time can have a big influence on cache hit ratios and the resulting link loads.

For the models in Papers C and D we rely to a large extent on statistics from traditional scheduled TV. In Paper E we study access patterns in a real TV-on-Demand system over four months. We study user behaviour and program popularity and its impact on caching. We show how the popularity of TV-on-Demand programs changes over time. We see that the access pattern in a TV-on-Demand system very much depend on what type of content it offers. Furthermore, we find that the share of requests for the top most popular programs grows during prime time, and the change rate among them decreases. The cacheability is very high and the cache hit ratio increases during prime time when it is needed the most.

# Chapter 3

# Summary of the Papers and Their Contributions

This thesis is a collection of five papers. Papers A-B study different aspects of robust traffic engineering. Papers C-E investigate TV-on-Demand access patterns and the potential for caching. The papers are all published at refereed international conferences.

In Paper A we look at robust traffic engineering as an optimisation problem. In Paper B we build upon the work in Paper A by applying the ideas to the legacy routing protocols OSPF and IS-IS. We study search heuristics for finding weight-settings, and evaluate how different cost functions manage to handle faults in input traffic data due to traffic hotspots.

In Paper C we use an empirical IPTV workload model to simulate IPTV distribution with time-shift and investigate the benefit of introducing a local cache closer to the TV subscribers. In Paper D we extend the work by looking at how TV program popularity changes over time. For the simulations in Paper C and D we use TV schedules and statistics from linear broadcast TV. In Paper E we analyse logs from a large TV-on-Demand system over four months.

## 3.1 Paper A: A Multi Path Routing Algorithm for IP Networks Based on Flow Optimisation

Henrik Abrahamsson, Juan Alonso, Bengt Ahlgren, Anders Andersson and Per Kreuger. A Multi Path Routing Algorithm for IP Networks Based on Flow Optimisation. In *Proceedings of Third COST 263 International Workshop on Quality of Future Internet Services (QoFIS 2002)*, Zurich, Switzerland, October 2002.

**Summary:**
Intra-domain routing in the Internet normally uses a single shortest path to forward packets towards a specific destination with no knowledge of traffic demand. We present an intra-domain routing algorithm based on multi-commodity flow optimisation which enables load sensitive forwarding over multiple paths. It is neither constrained by weight-tuning of legacy routing protocols, such as OSPF, nor requires a totally new forwarding mechanism, such as MPLS. These characteristics are accomplished by aggregating the traffic flows destined for the same egress into one commodity in the optimisation and using a hash based forwarding mechanism. The aggregation also results in a reduction of computational complexity which makes the algorithm feasible for on-line load balancing. Another contribution is the optimisation objective function which allows precise tuning of the tradeoff between load balancing and total network efficiency.

**Contribution:**
There are two contributions in this paper: the modelling of the problem as an optimisation problem, and the definition of an optimisation objective function for $l$-balanced solutions. In the modelling of the optimisation problem we aggregate all traffic destined for a certain egress into one commodity in a multi-commodity flow optimisation. It is this definition of a commodity that both makes the computation tractable, and the forwarding simple.

$L$-balanced solutions allows the network operator to choose a maximum desired link utilisation level. The optimisation will then find the most efficient solution, if it exists, satisfying the link level constraint. Our objective function thus enables the operator to control the trade-off between minimising the network utilisation and balancing load over multiple paths.

**My contribution:**
This is joint work with Bengt Ahlgren, Juan Alonso, Anders Gunnar and Per Kreuger. Juan Alonso did most of the mathematical work for this paper. In

discussion with Juan I contributed to the idea of only looking at the destination of the traffic when formulating the optimisation problem. I co-authored the paper.

## 3.2 Paper B: Robust Traffic Engineering using L-balanced Weight-Settings in OSPF/ISIS

Henrik Abrahamsson and Mats Björkman. Robust Traffic Engineering using L-balanced Weight-Settings in OSPF/ISIS. In: *Sixth International Conference on Broadband Communications, Networks, and Systems (BROADNETS 2009)*, September 2009, Madrid, Spain.

**Summary:**
The focus of this work is on robust traffic engineering for the legacy routing protocols OSPF and IS-IS. The idea is to use the *l*-balanced solutions proposed in Paper A to make sure that there are enough spare capacity on all links to handle sudden hotspots and traffic shifts. Search heuristics are used to find the set of weights that avoid loading any link to more than *l* and the resulting routings are evaluated using real topologies and traffic scenarios.

**Contribution:**
The contributions are the idea of *l*-balanced weight-settings for robust traffic engineering, the search heuristics for finding such weight-settings, and the evaluation of how different cost functions (including *l*-balanced) manage to handle faults in input traffic data due to traffic hotspots.

**My contribution:**
The idea of using the *l*-balanced solution for robust weight-settings was mine. I implemented the search heuristics and did the evaluations and wrote most of the paper.

## 3.3 Paper C: Simulation of IPTV caching strategies

Henrik Abrahamsson and Mats Björkman. Simulation of IPTV caching strategies. In: *International Symposium on Performance Evaluation of Computer*

*and Telecommunication Systems (SPECTS'10)*, 11-14 July 2010, Ottawa, Canada

**Summary:**
In this paper we use an empirical IPTV workload model to simulate IPTV distribution with time-shift and investigate the benefit of introducing a local cache closer to the TV subscribers. The simulations are based on real TV schedules, and statistics about TV program popularity and viewer activity. We simulate a large number of TV viewers that, when active, request scheduled or on-demand programs and we investigate the resulting bandwidth requirements on the down link for different cache sizes and caching strategies.

**Contribution:**
The contributions of this paper are: We present an empirical IPTV workload model. We simulate a realistic scenario for IPTV distribution and compare the Least Recently Used (LRU) and Least Frequently Used (LFU) caching strategies. We show that time-shifted TV can be very capacity demanding and that considerable amounts of bandwidth can be saved by caching the most popular programs closer to the viewers.

**My contribution:**
I designed and implemented the simulator, did the evaluations and wrote most of the paper.

## 3.4 Paper D: Caching for IPTV distribution with time-shift

Henrik Abrahamsson and Mats Björkman. Caching for IPTV distribution with time-shift. In: *International conference on Computing, Networking & Communications (ICNC'13)*, 28-31 January 2013, San Diego, USA.

**Summary:**
In this paper we simulate TV distribution with time-shift and investigate what impact TV program popularity, program set size, cache replacement policy and other factors have on the caching efficiency. The simulation results show that introducing a local cache close to the viewers significantly reduces the network load from TV on-demand services. By caching 4% of the program volume we can decrease the peak load during prime time by almost 50%. We also show that the TV program type and how program popularity changes over time can

have a big influence on cache hit ratios and the resulting link loads.

**Contribution:**
In this paper we extend the work in Paper C by looking at how TV program popularity changes over time. Many programs such as news programs and weather forecasts quickly become outdated and lose their popularity when available on-demand. Other programs, typically drama TV-shows, retain interest from some viewers even a long time after their first release and initial peak in popularity. We show that the TV program type and how program popularity changes over time can have a big influence on cache hit ratio and the resulting link loads.

**My contribution:**
I did the analysis of program popularity, implemented the simulator, did the evaluations and wrote most of the paper.

## 3.5 Paper E: Program popularity and viewer behaviour in a large TV-on-Demand system

Henrik Abrahamsson and Mattias Nordmark. Program popularity and viewer behaviour in a large TV-on-Demand system. In: *Internet Measurement Conference (IMC'12)*, 14-16 November 2012, Boston, USA.

**Summary:**
In this paper we analyse the access patterns in a large TV-on-Demand system and study the potential for caching. We characterize access patterns for different program categories, we show how program popularity changes over time and how this differs between different program types. We then use the request sequence in the data set for trace-driven simulation and study cache hit ratios for different cache sizes, cache replacement policies and population sizes.

**Contribution:**
Our contribution in this paper is three-fold. As a first-order result, we provide reconfirmation of known observations with an independent dataset. We demonstrate that there is a small set of programs that account for a large part of the requests. The program popularity conforms with the Pareto principle, or 80-20 rule. The demand follows a diurnal and weekly pattern, and there are large peaks in demand on Friday and Saturday evenings that need to be handled.

Second, we provide systematic evidence of TV-on-Demand access pattern characteristics that are intuitive yet unconfirmed in the literature. We show that news programs have a very short lifespan and are often only requested for a few hours, childrens programs are top ranked in the mornings and early evenings, and movie rentals are concentrated over weekends.

Finally, we also provide novel insights into access patterns that have not been reported previously to the best of our knowledge. We show how the popularity of TV-on-Demand programs changes over time. We see that the access pattern in a TV-on-Demand system very much depend on what type of content it offers. Furthermore, we find that the share of requests for the top most popular programs grows during prime time, and the change rate among them decreases. The cacheability is very high and the cache hit ratio increases during prime time when it is needed most.

**My contribution:**
I did the analysis with help from Mattias Nordmark. I did the simulations and I wrote the paper.

# Chapter 4

# Related Work

## 4.1 Traffic engineering in IP networks

Many different approaches for dynamic routing and traffic engineering have been proposed and used in telecommunication [45] and computer networks. For instance, the early ARPANET routing algorithms were based on measured link delay but had problems with traffic shifts and oscillations [46, 47].

The IETF Network Working Group presented a taxonomy of Internet traffic engineering methods in RFC3272 [48] in 2002. But for much of the traffic engineering research at that time the existing routing protocols were fixed. The challenge was to find configurations that adapted the routing to the current traffic situation. Traffic engineering by finding a suitable set of weights in OSPF/IS-IS is now a well studied area of research and it is described in textbooks in the area [25, 49]. When we in Paper B revisited the weight setting approach to traffic engineering we were most inspired by the pioneering works by Fortz and Thorup [50, 51] and Ramakrishnan and Rodrigues [52], in that we use a piece-wise linear cost function and search heuristics to find suitable weight settings.

Several studies [50, 53, 54, 55] have shown that even though we limit the routing of traffic to what can be achieved with weight-based ECMP shortest paths, and not necessarily the optimal weights but those found by search heuristics, it often comes close to the optimal routing for real network scenarios. How the traffic is distributed in the network very much depends on the objectives, usually expressed as a cost function, in the optimisation. An often proposed objective function is described by Fortz and Thorup [50]. Here the sum of the

cost over all links is considered and a piece-wise linear increasing cost function is applied to the flow on each link. The basic idea is that it should be cheap to use a link with small utilisation while using a link that approaches 100% utilisation should be heavily penalised. The *l*-balanced cost function used in Papers A and B is similar in that it uses a piecewise linear cost function to obtain desirable solutions. Additionally, it gives the operator the opportunity to set the maximum wanted link utilisation. Cost functions for traffic engineering is further investigated by Balon et al. [56]

Paper B added to existing work on weight settings by focusing on robustness and the objective of achieving a controlled spare capacity for handling unpredictable traffic shifts. For robust traffic engineering much of the focus has been on handling multiple traffic matrices and traffic scenarios [51, 57, 58, 59, 60, 61] and handling the trade-off between optimising for the common case or for the worst case. Nucci et al. [62] investigate link weight assignments that take into account SLA requirements and link failures. Xu et al. [63] describe a method to jointly solve the flow optimisation and the link-weight approximation using a single formulation resulting in a more efficient computation. Their method can also direct traffic over non-shortest paths with arbitrary percentages. Their results should also be directly applicable to our problem of providing robustness to changes, by just substituting their piece-wise linear cost function with our cost function. In a continuation on this work Xu et al. [64] propose a new link-state routing protocol. The protocol splits traffic over multiple paths with an exponential penalty on longer paths and achieves optimal traffic engineering while retaining the simplicity of hop-by-hop forwarding.

There are also several proposed traffic engineering protocols such as MATE [65], TeXCP [66] and REPLEX [67], that can balance traffic over several paths between ingress and egress nodes in the network, for instance by using MPLS [26]. Recently, much research focus has also been on OpenFlow and Software-defined Networking (SDN) with the possibility of fine-grained, flow-based management and control, and the separation of control plane and data plane functionality [27, 30, 31, 32, 33].

The advantage of optimising the weights in OSPF and IS-IS is of course easy deployment of the traffic engineering mechanism. However, the disadvantage is the difficulties and constraints imposed by using legacy routing. The general problem of finding the best way to route traffic through a network can be mathematically formulated as a multi-commodity flow (MCF) optimisation problem. In Paper A we present a routing algorithm based on multi-commodity flow optimisation. By aggregating the traffic flows destined for the same egress into one commodity in the optimisation we reduce the computational complex-

ity. The same approach was later used for instance by Sridharan et al. [68] and Fu et al. [69]. MCF optimisation is also used by many other research groups to address traffic engineering problems including [50, 70]. See also the book by Pioro and Medhi [49] and references therein.

## 4.2 Access patterns and potential for caching for TV and video on-demand

The recent growth and popularity of IPTV services have led to an increasing interest from researchers to measure and model IPTV viewing behavior. Cha et al. [71] present an extensive study of viewing behavior including channel popularity and channel switching in an operational IPTV network. Ramos et al. [72] present work on constructing an IPTV workload model capturing the way viewers change channels and watch live TV. Yu et al [73] study user activity and channel zapping in a municipal network. Qiu et al. model TV channel popularity [74] and user activities [75] in a large IPTV system and present the SimulWatch workload generator. These studies are similar to ours in that they model IPTV viewer behavior – but they study traditional live TV, and model channel popularity and not the popularity of individual programs. In Papers C and D we also simulate TV channels but our focus is on investigating time-shifted TV and the potential for caching. For this the popularity of individual programs is a fundamental part of the model. In this sense our work is closer to studies of traditional VoD systems.

Yu et al. [76] present a large measurement study of the Chinese PowerInfo Video-on-Demand system. This work is similar to ours in that they investigate many aspects of user behaviour and content access patterns. The PowerInfo system is a traditional VoD system. The videos in the library are old TV shows and movies and there are usually only a few new movies introduced to the system per day. This is different from the TV-on-Demand system that we study where there is a large inflow of new programs from the TV-schedule, time-shifted viewing, and programs with a very short life-span. Our work in Paper E is also different in other aspects in that we investigate how the access pattern depend on genre, we study cacheability and use trace-based simulation to investigate what impact the access patterns have on caching.

There are many other interesting studies of VoD systems and video popularity. Griwodz et al. [77] model long-term popularity of videos on the time scale of days based on VHS rental statistics. Lou et al. [78] give examples of the popularity evolution of video files from a Chinese television station. Tang

et al. [79] analyse and model many aspects of media server access. Avramova et al. [80] model the popularity evolution of TV-on-demand and video traces. Dan and Carlsson [81] measure and analyse BitTorrent content popularity. Guo et al. [82] study the probability distributions of Internet media workloads and analyse caching using a mathematical model. Yin et al. [83] study live VoD workloads from the 2008 Beijing Olympics. There are also many studies of Youtube and user generated videos [84, 85, 86, 87]. Szabo and Huberman [88] predict the long-term popularity of online content at Digg and Youtube based on early measurements of user accesses. Much research and many measurement studies have also focused on peer-assisted techniques for TV and VoD including [89, 90, 91, 92, 93, 94]. Ager et al. [95] study the cacheability for HTTP- and P2P-based applications.

Gopalakrishnan et al. [96] study user behaviour in a large IPTV system. This is similar to our work but their focus is on modeling the interactive user behaviour in an IPTV environment, including how users fast-forward, pause and rewind to control their viewing.

In Papers C and D we use an empirical IPTV workload model to simulate IPTV distribution and study caching. The simulations are based on real TV schedules, and statistics about TV program popularity and viewer activity. In Paper E we use trace-driven simulation, and utilize the sequence of requests in logs from a real TV-on-Demand system. There is also a lot of related work that use analytical models and simulations to study the performance of caching including [39, 40, 41, 42, 97, 98]. These studies have a more theoretical approach and is in this sense complementary to our work.

Seen in a broader perspective, a vast amount of research has been done on caching architectures, algorithms and protocols for instance for web, video and content distribution networks, as described in Section 1.4.2.

Another important issue for traffic and cache management is the interaction between traditional traffic engineering and content distribution in operator networks. What techniques and optimisations are possible here depend on the level of knowledge and control that the operator can have of the content distributed [99, 100, 101].

# Chapter 5

# Conclusions and Future Work

## 5.1 Conclusions

The Internet traffic volume continues to grow at a great rate, now pushed on by video and TV distribution in the networks. Increasing traffic volumes and the introduction of delay and loss sensitive services makes it crucial for operators to understand and manage the traffic situation in the network. More traffic also necessitate upgrades of network equipment and new investments for operators, and keep up-to-date the question of over-dimensioning network capacity versus using mechanisms for better handling the traffic.

This thesis deals with two approaches for avoiding network overload: traffic engineering and caching. We study traffic engineering mechanisms for adapting the routing to the current traffic situation and to steer traffic away from overloaded links. We study TV-on-Demand access patterns and the possible benefits of using caching mechanisms to avoid loading links with repeated transfers of popular content.

This thesis proposes $l$-balanced routings as a way for an operator to handle traffic variability and uncertainties in input traffic data. An $l$-balanced routing algorithm based on multi-commodity flow optimisation was presented in Paper A. A heuristic search method for finding $l$-balanced weight settings for the legacy routing protocols OSPF and IS-IS was presented in Paper B. $L$-balanced routing gives the operator possibility to apply simple rules of thumb for con-

trolling the maximum link utilisation and control the amount of spare capacity needed to handle sudden traffic variations. It gives more controlled traffic levels than other cost functions and more efficient routing for low traffic loads when there is no need to spread traffic over longer paths. The evaluation in Paper B shows that the search and the resulting weight settings work well in real network scenarios.

In Papers C-E we study TV-on-Demand access pattern and the potential for caching. We observe that there is a small set of programs that account for a large part of the requests. The program popularity conforms with the Pareto principle, or 80-20 rule. The demand follows a diurnal and weekly pattern, and there are large peaks in demand on Friday and Saturday evenings that need to be handled.

The popularity of rental movies, news, and TV shows changes over time in very different ways. News programs are often only requested for a few hours, movies are popular for months and increase in rank during weekends, TV shows increase in rank when the next episode is shown, and children's programs are top ranked in the mornings and early evenings. This means that programs jumps in and out of the top list of most popular programs. This can have implications for the choice of caching strategy. It is important to have the right programs in the cache in the evenings when the total demand is the highest. Another conclusion is that the access pattern in a TV-on-Demand system very much depend on what type of content it offers. We also observe that the request pattern for different episodes of the same show, and for programs within the same genre, often are very similar.

Another conclusion, from studying the cache friendliness of the TV-on-Demand workload, is that the potential for caching is high. The cacheability is very high, and in many scenarios the cache hit ratio with basic replacement policies is above 50% when caching 5% of the daily demand. We also observe that the hit ratio increases during prime time. The share of requests for the top most popular programs grows during prime time, and the change rate among them decreases.

## 5.2 Future work

It is an interesting time to work on issues related to television and video distribution over IP networks and the Internet. It is an area with fast development. Even with a view limited to Sweden in autumn 2012, as of this writing, a lot of things happen on many levels.

The way we are watching TV is slowly changing towards IP distributed television with more opportunities to choose what we want to watch, and when and where we watch TV. This thesis studies TV-on-Demand access patterns and the impact on caching. But the media consumption pattern is a moving target, it changes when the TV and video services evolve, and for future work there is a need to continuously study user behaviour and access patterns.

There is a trend towards start-over TV and TV-on-Demand where the viewer can choose to watch broadcast programs from the beginning or later after its scheduled time. Perhaps this is the beginning of a development in which the TV schedule becomes more a part of a recommendation system or a personalized playlist with a mix of live and pre-recorded content.

The devices are changing. More and more TV is watched on Internet connected smart TVs, and on smaller devices such as phones and tablets.

Another trend is that the TV and video market is changing. New players appear and compete with existing services. Many telecom and broadband operators have become TV distributors and offer new TV services in their own networks. HBO and Netflix, American providers of on-demand Internet streaming media, were launched in Sweden during the autumn 2012.

Traditional TV broadcasters are now also starting to distribute the scheduled TV via the web. It is also common with web exclusive content. One example of TV content that is often sent over the Internet today is sports events. SVT, the Swedish public service television company, showed 1600 hours from the London Olympics in two traditional broadcast channels and in six web channels [102]. A lot of the content was exclusively shown on the the web. But the change in viewing behaviour is still at an early stage and it takes time. The vast majority of viewing continues to be via traditional broadcast.

Television is a big thing. Although it is so commonplace that we might not think about it. When the way we are watching TV changes it can have a big impact on the distribution networks. In Sweden more than 70% of the population watch something on television on an average day, more than 40% of the people are watching TV during primetime, and individual TV shows can sometimes assemble 30-45% of the population [103]. If the TV viewing shifts from traditional broadcast to on-demand, personalized viewing on mobile devices, then it also gives rise to interesting future technical challenges.

The increasing demand for high-bandwidth streaming media services, both operator managed and OTT services, puts a big load on the networks. Caching seems to be a promising part of the solution. But there are many open issues for future work about caching, for instance what should be stored and on what level in the network should the caching be done.

In Papers C-E in this thesis we study many aspects of the access patterns in TV-on-Demand systems. We look at the cache friendliness of the workload in terms of cacheability and hit ratios for basic replacement policies. An immediate future work is to try to design and evaluate a caching strategy that is customized for the TV-on-Demand access patterns and investigate the extent to which it can reduce the network load.

When studying the cache friendliness of the request stream in Papers C-E we used the basic LRU and LFU cache replacement policies. With these the last requested program is always cached and the choice of what to evict from the cache is between the least recently and the least frequently requested program. A more advanced system could use more knowledge about access patterns and program popularity to decide what program to put in the cache and what program to evict.

One such strategy could be to keep track of all programs in the system, also those that are not currently in the cache. One could monitor the popularity by counting requests, let the programs age over time and for each program keep a value that describes the probability that it will be requested. There are a number of observations about the access patterns in this thesis that can be useful for such an informed caching strategy:

**Give preference to new programs** The broadcast of the traditional TV schedule has a marketing effect and with time-shifted TV ongoing scheduled programs immediately get a lot of requests. Some programs, like TV-news, also have a very short life-span. The value of a program should not have to be built up by requests over a long time.

**Categorize programs by genre to predict change in popularity over time** We see in Papers D and E that the access pattern very much depends on the type of program. A news program that is top-ranked the first evening age quickly and has a very low probability for being requested the next evening. A rental movie however is popular for months and increase in rank during weekends. By categorizing programs by genre the probability for future requests can be predicted. The categorization of programs can also be more detailed. The request patterns for different episodes of the same show are often surprisingly similar. For a new episode of a show it is a reasonable assumption that the popularity of the program will change over time in a way similar to that of the previous episodes.

**Focus on prime time** The value of a program should reflect the probability that it will be requested during prime time. There are large peaks in

demand in the evenings and at the weekends that need to be handled. If caching is used to limit the maximum link load then it is essential to have the right programs in the cache on Friday and Saturday evenings. There are program like cartoons that are top-ranked in the mornings and early evenings that probably should never be in the cache.

The observations and the predictions outlined above can be used to optimise the caching performance. However, the basic monitoring of request frequency is still needed as a basis, and to handle unexpected changes and sudden peaks in program demand for instance due to large news events.

In Papers C-E we see that the cacheability and cache hit ratios for the TV-on-Demand workload are high even for small populations. Introducing a comparatively small local cache could significantly reduce the peak link loads. But for operators the monetary cost (both the capital expenditures and operational costs) of introducing memory into the network versus providing the bandwidth needed is essential. This is an important aspect to consider in future work.

# Bibliography


[1] Measuring the Information Society 2012. International Telecommunication Union, report, October 2012.

[2] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. Internet Inter-Domain Traffic. In *Proceedings of the ACM SIGCOMM'10*, New Delhi, India, 2010.

[3] Wikipedia. Tier 1 network. http://en.wikipedia.org/wiki/Tier_1_network, October 2012.

[4] Bernhard Ager, Nikolaos Chatzis, Anja Feldmann, Nadi Sarrar, Steve Uhlig, and Walter Willinger. Anatomy of a Large European IXP. In *Proceedings of ACM SIGCOMM'12*, Helsinki, Finland, 2012.

[5] Juan Camilo Cardona Restrepo and Rade Stanojevic. A History of an Internet eXchange Point. *ACM SIGCOMM Computer Communication Review*, 42(2), April 2012.

[6] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The Akamai Network: A Platform for High-Performance Internet Applications. *ACM SIGOPS Operating Systems Review*, 44(3), July 2010.

[7] Tom Leighton. Improving Performance of the Internet. *Communications of the ACM*, 52(2), February 2009.

[8] Ao-Jan Su, David R. Choffnes, Aleksandar Kuzmanovic, and Fabián E. Bustamante. Drafting behind akamai (travelocity-based detouring). In *Proceedings of SIGCOMM '06*, Pisa, Italy, 2006.

[9] Ruben Torres, Alessandro Finamore, Jin Ryong Kim, Marco Mellia, Maurizio M. Munaf, and Sanjay G. Rao. Dissecting Video Server Selection Strategies in the YouTube CDN. In *Proceedings of 31st International Conference on Distributed Computing Systems (ICDCS'11)*, June 2011.

[10] Vijay Kumar Adhikari, Sourabh Jain, Yingying Chen, and Zhi-Li Zhang. Vivisecting YouTube: An Active Measurement Study. In *Proceedings of IEEE INFOCOM'12, Mini-conference*, Orlando, USA, March 2012.

[11] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery. In *Proceedings of IEEE INFOCOM'12*, Orlando, USA, March 2012.

[12] Announcing the Netflix Open Connect Network. Netflix blog, http://blog.netflix.com/2012/06/announcing-netflix-open-connect-network.html, June 2012.

[13] Google and Netflix Make Land Grab On Edge Of Internet. Wired, http://www.wired.com/wiredenterprise/2012/06/cdn/, June 2012.

[14] Alexandre Gerber and Robert Doverspike. Traffic Types and Growth in Backbone Networks. In *Proceedings of OFC/NFOEC*, March 2011.

[15] Kenjiro Cho, Kensuke Fukuda, Hiroshi Esaki, and Akira Kato. Observing Slow Crustal Movement in Residential User Traffic. In *Proceedings ACM CoNEXT'08*, Madrid, Spain, December 2009.

[16] Netnod. On–line: http://www.netnod.se/ix/statistics.

[17] Jie Li, Andreas Aurelius, Viktor Nordell, Manxing Du, Åke Arvidsson, and Maria Kihl. A five year perspective of traffic pattern evolution in a residential broadband access network. In *Proceedings of Future Network & Mobile Summit 2012*, Berlin, Germany, July 2012.

[18] Gregor Maier, Anja Feldmann, Vern Paxson, and Mark Allman. On Dominant Characteristics of Residential Broadband Internet Traffic. In *Proceedings of the 9th ACM SIGCOMM Internet measurement conference (IMC'09)*, Chicago, Illinois, USA, 2009.

[19] Pierre Borgnat, Guillaume Dewaele, Kensuke Fukuda, Patrice Abry, and Kenjiro Cho. Seven Years and One Day: Sketching the Evolution of Internet Traffic. In *Proceedings on INFOCOM'09*, 2009.

[20] Wolfgang John, Sven Tafvelin, and Tomas Olovsson. Trends and differences in connection-behavior within classes of internet backbone traffic. In *Proceedings of the 9th international conference on Passive and active network measurement (PAM'08)*, Cleveland, USA.

[21] Subhabrata Sen and Jia Wang. Analyzing peer-to-peer traffic across large networks. *IEEE/ACM Transactions on Networking*, 12(2):219–232, April 2004.

[22] G. J. Miller K. Thompson and R. Wilder. Wide-area traffic patterns and characteristics (extended version). *IEEE Network*, 11(6):10–23, 1997.

[23] Sandvine. Global Internet Phenomena Report, 1H 2012. http://www.sandvine.com/news/global_broadband_trends.asp, 2012.

[24] Bernard Fortz, Jennifer Rexford, and Mikkel Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communication Magazine*, 40, October 2002.

[25] J. Rexford. Route optimization in IP networks. In Mauricio G.C. Resende and Panos M. Pardalos, editors, *Handbook of Optimization in Telecommunications*. Springer Science+Business Media, 2006.

[26] Multiprotocol label switching working group . http://datatracker.ietf.org/wg/mpls/charter/.

[27] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Computer Communication Review*, 38(2):69–74, April 2008.

[28] Openflow @ google. Keynote by Urs Hoelzle, Open Networking Summit 2012, http://www.opennetsummit.org.

[29] Going With the Flow: Googles Secret Switch to the Next Wave of Networking. *Wired*, April 2012. http://www.wired.com/wiredenterprise/2012/04/going-with-the-flow-google/.

[30] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. NOX: Towards an Operating System for Networks. *SIGCOMM Computer Communication Review*, 38(3):105–110, July 2008.

[31] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, and Scott Shenker. Onix: A Distributed Control Platform for Large-scale Production Networks. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation (OSDI'10)*, Vancouver, Canada, 2010.

[32] Minlan Yu, Jennifer Rexford, Michael J. Freedman, and Jia Wang. Scalable Flow-Based Networking with DIFANE. In *Proceedings of the ACM SIGCOMM'10*, New Delhi, India, 2010.

[33] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. DevoFlow: Scaling Flow Management for High-Performance Networks. In *Proceedings of the ACM SIGCOMM'11*, Toronto, Canada, 2011.

[34] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of IEEE INFOCOM*, 1999.

[35] Jia Wang. A Survey of Web Caching Schemes for the Internet. *ACM SIGCOMM Computer Communication Review*, 29:36–46, 1999.

[36] Alec Wolman, Geoffrey M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On the scale and performance of cooperative Web proxy caching. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP '99)*, 1999.

[37] Abdullah Balamash and Marwan Krunz. An Overview of Web Caching Replacement Algorithms. *IEEE Communication surveys*, 6:44–56, 2004.

[38] Jiangchuan Liu and Jianliang Xu. Proxy Caching for Media Streaming Over the Internet. *IEEE Communications Magazine*, 42:88–94, 2004.

[39] Sem Borst, Varun Gupta, and Anwar Walid. Distributed caching algorithms for content distribution networks. In *Proceedings of INFOCOM'10*, San Diego, USA, 2010.

[40] D. De Vleeschauwer, Z. Avramova, S. Wittevrongel, and H. Brueel. Transport Capacity for a Catch-up Television Service. In *Proceedings of EuroITV'09*, pages 161–170, Leuven, Belgium, June 2009.

[41] Danny De Vleeschauwer and Koenraad Laevens. Performance of caching algorithms for IPTV on-demand services. *IEEE Transactions on broadcasting*, 55:491 – 501, 2009.

[42] T. Wauters, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, and E. Six. Co-operative Proxy Caching Algorithms for Time-Shifted IPTV Services. In *Proceedings of 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pages 379–386, Dubrovnik, Croatia, September 2006.

[43] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A Survey of Information-Centric Networking. *IEEE Communications Magazine*, 50(7), July 2012.

[44] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT'09)*, Rome, Italy, December 2009.

[45] Gerald Ash. *Dynamic Routing in Telecommunications Networks*. McGraw-Hill, 1997.

[46] John M. McQuillan, Ira Richer, and Eric C. Rosen. The New Routing Algorithm for the ARPANET. *IEEE Transactions on communications*, 28(5), May 1980.

[47] A. Khanna and J. Zinky. The revised ARPANET routing metric. In *Symposium proceedings on Communications architectures & protocols*, SIGCOMM '89, 1989.

[48] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of Internet traffic engineering. Internet RFC 3272, May 2002.

[49] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann, 2004.

[50] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000*, pages 519–528, Israel, March 2000.

[51] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, May 2002.

[52] K.G. Ramakrishnan and M.A. Rodrigues. Optimal routing in shortest path data networks. *Lucent Bell Labs Technical Journal*, 6(1), 2001.

[53] A. Gunnar, H. Abrahamsson, and M. Söderqvist. Performance of Traffic Engineering in Operational IP-Networks: An Experimental Study. In *Proceedings of 5th IEEE International Workshop on IP Operations and Management*, Barcelona, Spain, October 2005.

[54] A. Sridharan, R. Guérin, and C. Diot. Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks. In *IEEE Infocom*, San Francisco, March 2003.

[55] David Applegate and Edith Cohen. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs. In *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[56] S. Balon, F. Skivee, and G. Leduc. How Well Do Traffic Engineering Objective Functions Meet TE Requirements? In *Proceedings of IFIP International Networking Conference*, Coimbra, Portugal, May 2006.

[57] C. Zhang, Z. Ge, J. Kurose, Y. Liu, and D. Townsley. Optimal Routing with Multiple Traffic Matrices: Tradeoffs between Average Case and Worst Case Performance. In *Proceedings of ICNP 2005*, Boston, USA, November 2005.

[58] C. Zhang, Y. Liu, W. Gong, J. Kurose, R. Moll, and D. Townsley. On Optimal Routing with Multiple Traffic Matrices . In *Proceedings of Infocom 2005*, Miami, USA, March 2005.

[59] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. Cope: Traffic engineering in dynamic networks. In *Proceedings of ACM SIGCOMM*, Pisa, Italy, September 2006.

[60] Anders Gunnar and Mikael Johansson. Robust routing under bgp reroutes. In *Proceedings of Globecom 2007*, Washington, DC, USA, 2007.

[61] Anders Gunnar. *Aspects of proactive traffic engineering in IP networks*. PhD thesis, KTH Royal Institute of Technology, 2011.

[62] Antonio Nucci, Supratik Bhattacharyya, Nina Taft, and Christophe Diot. IGP Link Weight Assignment for Operational Tier-1 Backbones. *IEEE/ACM Transactions on Networking*, 15(4), August 2007.

[63] Dahai Xu, Mung Chiang, and Jennifer Rexford. DEFT: Distributed exponentially-weighted flow splitting. In *IEEE Infocom*, Anchorage, Alaska, USA, May 6–12, 2007.

[64] Dahai Xu, Mung Chiang, and Jennifer Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. *IEEE/ACM Transactions on networking*, 19(6), December 2011.

[65] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS Adaptive Traffic Engineering. In *Proceedings of IEEE INFOCOM*, April 2001.

[66] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. Walking the tightrope: responsive yet stable traffic engineering. In *Proceedings of SIGCOMM '05*, Philadelphia, Pennsylvania, USA, 2005.

[67] Simon Fischer, Nils Kammenhuber, and Anja Feldmann. Replex: dynamic traffic engineering based on wardrop routing policies. In *Proceedings of the 2006 ACM CoNEXT conference*, CoNEXT '06, 2006.

[68] A. Sridharan, R. Guérin, and C. Diot. Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks. *IEEE/ACM Transactions on Networking*, 13, April 2005.

[69] Jing Fu, Peter Sjödin, and Gunnar Karlsson. Traffic engineering and routing in IP networks with centralized control. May 2008. IFIP Networking 2008.

[70] D. Mitra and K. G. Ramakrishnan. A Case Study of Multiservice, Multipriority Traffic Engineering Design for Data Networks. In *Proceedings of Globecom'99*, Brazil, 1999.

[71] Meeyoung Cha, Pablo Rodriguez, Jon Crowcroft, Sue Moon, and Xavier Amatriain. Watching Television Over an IP Network. In *Proceedings of Internet Measurement Conference(IMC)*, pages 71–84, Greece, October 2008.

[72] Fernando M.V. Ramos, Fei Song, Pablo Rodriguez, Richard Gibbens, Jon Crowcroft, and Ian H. White. Constructing an IPTV Workload Model. In *Proceedings of SIGCOMM, Poster session*, Barcelona, Spain, August 2009.

[73] Geng Yu, Tord Westholm, Maria Kihl, Iñigo Sedano, Andreas Aurelius, Christina Lagerstedt, and Per Ödling. Analysis and Characterization of IPTV user behavior. In *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB'09)*, 2009.

[74] Tongqing Qiu, Zihui Ge, Seungjoon Lee, Jia Wang, Qi Zhao, and Jun Xu. Modeling Channel Popularity Dynamics in a Large IPTV System. In *Proceedings of SIGMETRICS*, pages 275–286, Seattle, USA, June 2009.

[75] Tongqing Qiu, Zihui Ge, Seungjoon Lee, Jia Wang, Jun Xu, and Qi Zhao. Modeling User Activities in a Large IPTV System. In *Proceedings of Internet Measurement Conference*, pages 430–441, Chicago, USA, November 2009.

[76] H. Yu, D. Zheng, B. Zhao, and W. Zheng. Understanding User Behavior in Large-Scale Video-on-Demand Systems. In *Proceedings of EuroSys2006*, pages 333–344, Leuven, Belgium, 2006.

[77] C. Griwodz, Michael Bär, and L. Wolf. Long-term Movie Popularity Models in Video-on-Demand Systems or The Life of an on-Demand Movie. In *Proceedings of the fifth ACM international conference on Multimedia*, pages 349 – 357, Seattle, USA, 1997.

[78] J. Lou, Y. Tang, M. Zhang, and S. Yang. Characterizing User Behavior Model to Evaluate Hard Cache in Peer-to-Peer Based Video-on-demand Service. In *Proceedings of MMM'07*, pages 125–134, 2007.

[79] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat. Modeling and Generating Realistic Streaming Media Server Workloads. *Computer Networks*, 51:336–356, 2007.

[80] Z. Avramova, S. Wittevrongel, H. Bruneel, and D. Vleeschauwer. Analysis and Modeling of Video Popularity Evolution in Various Online Video Content Systems: Power Law versus Exponential Decay. In *Proceedings of International Conference on Evolving Internet*, 2009.

[81] György Dan and Niklas Carlsson. Power-law Revisited: A Large Scale Measurement Study of P2P Content Popularity. In *Proceedings of the 9th international conference on Peer-to-peer systems (IPTPS'10)*, San Jose, USA, April 2010.

[82] Lei Guo, Enhua Tan, Songqing Chen, Zhen Xiao, and Xiaodong Zhang. The stretched exponential distribution of internet media access patterns. In *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing (PODC'08)*, New York, USA, 2008.

[83] H. Yin, X. Liu, F. Qiu, N. Xia, C. Lin, H. Zhang, V. Sekar, and G. Min. Inside the Bird's Nest: Measurements of Large-Scale Live VoD from the 2008 Olympics. In *Proceedings of Internet Measurement Conference (IMC'09)*, USA, 2009.

[84] Y. Borghol, S. Mitra, S. Ardon, N. Carlsson, D. Eager, and A. Mahanti. Characterizing and Modeling Popularity of User-generated Videos. In *Proceedings of IFIP International Symposium on Computer Performance, Modeling, Measurements and Evaluation (PERFORMANCE)*, 2011.

[85] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon. Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems. *IEEE Transactions on Networking*, 17:1357–1370, October 2009.

[86] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. YouTube Traffic Characterization: A view From the Edge. In *Proceedings of ACM SIGCOMM Internet Measurement Conference*, San Diego, USA, October 2007.

[87] X. Kang, H. Zhang, G. Jiang, H. Chen, X. Meng, and K. Yoshihira. Measurement, Modeling, and Analysis of Internet Video Sharing Site Workload: A Case Study. In *Proceedings of IEEE Conference on Web Services*, 2008.

[88] G. Szabo and B. Huberman. Predicting the Popularity of Online Content. *Communications of the ACM*, 53, August 2010.

[89] C. Huang, J. Li, and K.W. Ross. Can Internet VoD be Profitable? In *Proceedings of ACM Sigcomm 2007*, pages 133–144, Kyoto, Japan, 2007.

[90] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang. Challenges, Design and Analysis of a Large-scale P2P-VoD System. In *Proceedings of ACM SIGCOMM 2008*, pages 375–388, Seattle, USA, August 2008.

[91] M. Cha, P. Rodriguez, S. Moon, and J. Crowcraft. On Next-Generation Telco-Managed P2P TV Architectures. In *Proceedings of International workshop on Peer-To-Peer Systems (IPTPS)*, 2008.

[92] X. Hei, C. Liang, J. Liang, Y. Liu, and K.W. Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9:1672–1687, December 2007.

[93] T. Silverstone and O. Fourmaux. Measuring P2P IPTV Systems. In *Proceedings of NOSSDAV'07*, Illinois, USA, 2007.

[94] Thomas Silverston, Olivier Fourmaux, Alessio Botta, Alberto Dainotti, Antonio Pescapé, Giorgio Ventre, and Kavé Salamatian. Traffic analysis of peer-to-peer IPTV communities. *Computer Networks*, 53(4):470–484, March 2009.

[95] Bernhard Ager, Fabian Schneider, Juhoon Kim, and Anja Feldmann. Revisiting Cacheability in Times of User Generated Content. In *Proceedings of 13th IEEE Global Internet Symposium*, San Diego, CA, USA, March 2010.

[96] V. Gopalakrishnan, R. Jana, K. Ramakrishnan, D. Swayne, and V. Vaishampayan. Understanding Couch Potatoes: Measurement and Modeling of Interactive Usage of IPTV at large scale. In *Proceedings of Internet Measurement Conference (IMC'11)*, 2011.

[97] Massimo Gallo, Bruno Kauffmann, Luca Muscariello, Alain Simonian, and Christian Tanguy. Performance Evaluation of the Random Replacement Policy for Networks of Caches. In *Proceedings of ACM SIGMETRICS'12*, London, England, UK, 2012.

[98] John Ardelius, Björn Grönvall, Lars Westberg, and Åke Arvidsson. On the Effects of Caching in Access Aggregation Networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, Helsinki, Finland, 2012.

[99] Wenjie Jiang, Rui Zhang-Shen, Jennifer Rexford, and Mung Chiang. Cooperative Content Distribution and Traffic Engineering in an ISP Network. In *Proceedings of ACM SIGMETRICS'09*, Seattle, USA, June 2009.

[100] Ingmar Poese, Benjamin Frank, Bernhard Ager, Georgios Smaragdakis, and Anja Feldmann. Improving Content Delivery Using Provider-aided Distance Information. In *Proceedings of Internet Measurement Conference (IMC'10)*, Melbourne, Australia, November 2010.

[101] Ingmar Poese, Benjamin Frank, Georgios Smaragdakis, Steve Uhlig, Anja Feldmann, and Bruce Maggs. Enabling Content-aware Traffic Engineering. *ACM SIGCOMM Computer Communication Review*, 42(5):21–28, October 2012.

[102] Datorn ger OS-tittandet ett lyft. *Dagens Nyheter*, 24 July 2012. http://www.dn.se/sport/datorn-ger-os-tittandet-ett-lyft.

[103] Mediamätning i Skandinavien (MMS). On–line: http://www.mms.se.

# II

# Included Papers

# Chapter 6

# Paper A: A Multi Path Routing Algorithm for IP Networks Based on Flow Optimisation

Henrik Abrahamsson, Bengt Ahlgren, Juan Alonso, Anders Andersson, and Per Kreuger. In *Proceedings of the Third International Workshop on Quality of Future Internet Services (QoFIS)*, Zürich, Switzerland, October 2002.

**Abstract**

Intra-domain routing in the Internet normally uses a single shortest path to forward packets towards a specific destination with no knowledge of traffic demand. We present an intra-domain routing algorithm based on multi-commodity flow optimisation which enable load sensitive forwarding over multiple paths. It is neither constrained by weight-tuning of legacy routing protocols, such as OSPF, nor requires a totally new forwarding mechanism, such as MPLS. These characteristics are accomplished by aggregating the traffic flows destined for the same egress into one commodity in the optimisation and using a hash based forwarding mechanism. The aggregation also results in a reduction of computational complexity which makes the algorithm feasible for on-line load balancing. Another contribution is the optimisation objective function which allows precise tuning of the tradeoff between load balancing and total network efficiency.

## 6.1 Introduction

As IP networks are becoming larger and more complex, the operators of these networks gain more and more interest in *traffic engineering* [1]. Traffic engineering encompasses performance evaluation and performance optimisation of operational IP networks. An important goal with traffic engineering is to use the available network resources more efficiently for different types of load patterns in order to provide a better and more reliable service to customers.

Current routing protocols in the Internet calculate the shortest path to a destination in some metric without knowing anything about the traffic demand or link load. Manual configuration by the network operator is therefore necessary to balance load between available alternate paths to avoid congestion. One way of simplifying the task of the operator and improve use of the available network resources is to make the routing protocol sensitive to traffic demand. Routing then becomes a flow optimisation problem.

One approach taken by others [2, 3, 4] is to let the flow optimisation result in a set of link weights that can be used by legacy routing protocols, e.g., open shortest path first (OSPF), possibly with equal cost multi-path (ECMP) forwarding. The advantage is that no changes are needed in the basic routing protocol or the forwarding mechanism. The disadvantage is that the optimisation is constrained by what can be achieved with tuning the weights. Another approach is to use MPLS [5], multi-protocol label switching, for forwarding traffic for large and long-lived flows. The advantage is that the optimisation is not constrained, but at the cost of more complexity in the routing and forwarding mechanisms.

Our goal is to design an optimising intra-domain routing protocol which is *not* constrained by weight-tuning, and which *can* be implemented with minor modifications of the legacy forwarding mechanism based on destination address prefix.

In this paper we present a routing algorithm for such a protocol based on multi-commodity flow optimisation which is both computationally tractable for on-line optimisation and also can be implemented with a near-legacy forwarding mechanism. The forwarding mechanism needs a modification similar to what is needed to handle the ECMP extension to OSPF.

The key to achieve this goal, and the main contribution of this paper, is in the modelling of the optimisation problem. We aggregate all traffic destined for a certain egress into one commodity in a multi-commodity flow optimisation. This reduces the number of commodities to at most $N$, the number of nodes, instead of being $N^2$ when the problem is modelled with one commodity for

each pair of ingress and egress nodes. As an example, the computation time for a 200 node network was in one experiment 35 seconds. It is this definition of a commodity that *both* makes the computation tractable, *and* the forwarding simple.

Another important contribution is the definition of an optimisation objective function which allows the network operator to choose a maximum desired link utilisation level. The optimisation will then find the most efficient solution, if it exists, satisfying the link level constraint. Our objective function thus enables the operator to control the trade-off between minimising the network utilisation and balancing load over multiple paths.

The rest of the paper is organised as follows. In the next section we describe the overall architecture where our optimising routing algorithm fits in. Section 6.3 presents the mathematical modelling of the optimisation problem. We continue with a short description of the forwarding mechanism in Sect. 6.4. After related work in Sect. 6.5 we conclude the paper.

## 6.2 Architecture

In this work we take the radical approach to completely replace the traditional intra-domain routing protocol with a protocol that is based on flow optimisation. This approach is perhaps not realistic when it comes to deployment in real networks in the near future, but it does have two advantages. First, it allows us to take full advantage of flow optimisation without being limited by current practise. Second, it results in a simpler overall solution compared to, e.g., the metric tuning approaches [2, 3, 4]. The purpose of taking this approach is to assess its feasibility and, hopefully, give an indication on how to balance flow optimisation functionality against compatibility with legacy routing protocols.

In this section we outline how the multi-commodity flow algorithm fits into a complete routing architecture. Figure 6.1 schematically illustrates its components. Flow measurements at all ingress nodes and the collection of the result are new components compared to legacy routing. The measurements continuously (at regular intervals) provide an estimate of the current demand matrix to the centralised flow optimisation. The demand matrix is aggregated at the level of all traffic from an ingress node destined for a certain egress node.

If a more fine-grained control over the traffic flows are desired, for instance to provide differentiated quality of service, a more fine-grained aggregation level can be chosen. This results in more commodities in the optimisation, which can be potential performance problem. One approach is to introduce two

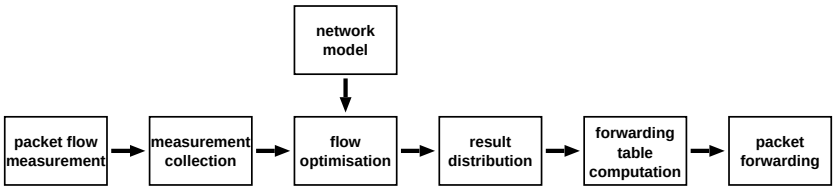



Figure 6.1: Routing architecture with flow optimisation.

levels in the optimisation, one with a longer time-scale for quality of service flows.

The demand matrix is input to the flow optimiser together with a model of the network. The result of the optimisation is a set of values $y_{ij}^t$, which encode how traffic arriving at a certain node ($i$), destined for a certain egress node ($t$) should be divided between the set of next hops ($j$). These values are used at each node together with a mapping between destination addresses and egress nodes to construct forwarding tables. Finally, the packet forwarding mechanism is modified to be able to distinguish packets destined for a certain egress node, and to forward along multiple paths toward those egresses.

The computation of the multi-commodity flow optimisation algorithm is inherently centralised. In this paper we also think of the computation as implemented in a central server. If a so-called bandwidth broker is needed or desired for providing a guaranteed quality of service, it is natural to co-locate it with optimisation. We however see the design of a distributed mechanism implementing flow optimisation as an important future work item.

The timescale of operation is important in an optimising routing architecture. There are several performance issues that put lower bounds on the cycle flow measurement–optimisation–new forwarding tables. The flow measurement need to be averaged over a long enough time to get sufficiently stable values. Our current research as well as others [6] indicate that the needed stability exists in real networks at the timescale of a few, maybe five to ten, minutes. Other performance issues are the collection of the flow measurements, the computation of the optimisation algorithm, and the distribution of the optimisation result. Our initial experiments indicate that a new optimisation cycle can be started in approximately each five minutes for typical intra-domain sizes.

An issue that we have identified is how to handle multiple egresses for a destination injected into the domain by BGP, the border gateway protocol. A straightforward way to solve this is to introduce additional virtual nodes in

the network to represent a common destination behind both egresses. This approach may however introduce a large number of additional nodes. This will need to be more carefully considered in the future.

## 6.3 Optimisation

The routing problem in a network consists in finding a path or multiple paths that send traffic through the network without exceeding the capacity of the links. When using optimisation to find such (multiple) paths, it is natural to model the traffic problem as a (linear) multi-commodity network flow problem (see, e.g., Ahuja et al. [7]), as many authors have done.

First, the network is modelled as a directed graph (this gives the topology, i.e., the static information of the traffic problem), and then the actual traffic situation (i.e., the dynamic part of the problem, consisting of the current traffic demand and link capacity) as a linear program. In modelling the network as a graph, a node is associated to each router and a directed edge to each directional link physically connecting the routers. Thus, we assume a given graph $G = (N, E)$, where $N$ is a set of nodes and $E$ is the set of (directed) edges. We will abuse language and make no distinction between graph and network, node and router, or edge and link.

Every edge $(i, j) \in E$ has an associated capacity $k_{ij}$ reflecting the bandwidth available to the corresponding link. In addition, we assume a given *demand matrix* $D = D(s, t)$ expressing the traffic demand from node $s$ to node $t$ in the network. This information defines the routing problem. In order to formulate it as a multi-commodity flow (MCF) problem we must decide how to model commodities. In the usual approach [7, 2, 8] commodities are modelled as source-destination pairs that are interpreted as "all traffic from source to destination". Thus, the set of commodities is a subset of the Cartesian product $N \times N$; consequently, the number of commodities is bounded by the square of the number of nodes. To reduce the size of the problem and speed-up computations, we model instead commodities as (only destination) nodes, i.e., a commodity $t$ is to be interpreted as "all traffic to $t$". Thus, our set of commodities is a subset of $N$ and, hence, there are at most as many commodities as nodes. The corresponding MCF problem can be formulated as follows:

$$\min \{f(y) \mid y \in P_{12}\} \qquad (MCF_{12})$$

where $y = (y_{ij}^t),$ for $t \in N, (i, j) \in E$, and $P_{12}$ is the polyhedron defined by

the equations:

$$\sum_{\{j|(i,j)\in E\}} y_{ij}^t - \sum_{\{j|(j,i)\in E\}} y_{ji}^t = d(i,t) \quad \forall i,t \in N \qquad (6.1)$$

$$\sum_{t\in N} y_{ij}^t \leq k_{ij} \quad \forall (i,j) \in E \qquad (6.2)$$

where

$$d(i,t) = \begin{cases} -\sum_{s\in N} D(s,t) & \text{if } i = t \\ D(i,t) & \text{if } i \neq t \end{cases} .$$

The variables $y_{ij}^t$ denote the amount of traffic to $t$ routed through the link $(i,j)$. The equation set (1) state the condition that, at intermediate nodes $i$ (i.e., at nodes different from $t$), the outgoing traffic equals the incoming traffic plus traffic created at $i$ and destined to $t$, while at $t$ the incoming traffic equals all traffic destined to $t$. The equation set (2) state the condition that the total traffic routed over a link cannot exceed the link's capacity.

It will also be of interest to consider the corresponding problem *without* requiring the presence of the equation set (2). We denote this problem ($MCF_1$). Notice that every point $y = (y_{ij}^t)$ in $P_{12}$ or $P_1$ represents a possible solution to the routing problem: it gives a way to route traffic over the network so that the demand is met and capacity limits are respected (when it belongs to $P_{12}$), or the demand is met but capacity limits are not necessarily respected (when it belongs to $P_1$). Observe that $y = (0)$ is in $P_{12}$ or in $P_1$ only in the trivial case when the demand matrix is zero.

A general linear objective function for either problem has the form $f(y) = \sum_{t,(i,j)} b_{ij}^t \; y_{ij}^t$. We will, however, consider only the case when all $b_{ij}^t = 1$ which corresponds to the case where all commodities have the same cost on all links. We will later use different objective functions (including non-linear ones) in order to find solutions with desired properties.

### 6.3.1 Desirable Solutions

In short, the solutions we consider to be desirable are those which are *efficient* and *balanced*. We make these notions precise as follows.

We use the objective function considered above, $f(y) = \sum_{t,(i,j)} y_{ij}^t$, as a measure of efficiency. Thus, given $y_1, y_2$ in $P_{12}$ or $P_1$, we say that $y_1$ is *more efficient* than $y_2$ if $f(y_1) \leq f(y_2)$. To motivate this definition, note that

whenever traffic between two nodes can be routed over two different paths of unequal length, $f$ will choose the shortest one. In case the capacity of the shortest path is not sufficient to send the requested traffic, $f$ will utilise the shortest path to 100% of its capacity and send the remaining traffic over the longer path.

Given a point $y = (y_{ij}^t)$ as above, we let $Y_{i,j} = \sum_{t \in N} y_{ij}^t$ denote the total traffic sent through $(i, j)$ by $y$. Every such $y$ defines a *utilisation* of edges by the formula $u(y, i, j) = Y_{ij}/k_{ij}$, and $u(y, i, j) = 0$ when $k_{ij} = 0$. Let $u(y)$ denote the maximum value of $u(y, i, j)$ where $(i, j)$ runs over all edges. Given an $\ell > 0$, we say that $y \in P_{12}$ (or $y \in P_1$) is *$\ell$-balanced* if $u(y) \leq \ell$. For instance, a solution is $(0.7)$-balanced if it never uses any link to more than 70 % of its capacity.

### 6.3.2 How to Obtain Desirable Solutions

Poppe et al. [8] have proposed using different linear objective functions in order to obtain traffic solutions that are desirable with respect to several criteria (including balance, in the form of minimising the maximum utilisation of edges). Fortz and Thorup [2, 3], on the other hand, considers a fixed piece-wise linear objective function (consisting of six linear portions for each edge) which makes the cost of sending traffic along an edge depend on the utilisation of the edge. By making the cost increase drastically as the utilisation approaches 100 %, the function favours balanced solutions over congested ones. As the authors express it, their objective function "provides a general best effort measure".

Our contribution is related to the above mentioned work in that we use different objective functions to obtain desirable solutions, and the functions are piece-wise linear and depend on the utilisation. In contrast, our work defines different levels of balance (namely, $\ell$-balance). For each such level, a simple piece-wise linear objective function consisting of two linear portions for each edge is *guaranteed* to find $\ell$-balanced solutions provided, of course, that such solutions exist. Moreover, the solution found is guaranteed to be more efficient than any other $\ell$-balanced solution.

Another distinctive feature of our functions is that they are defined through a uniform, theoretical "recipe" which is valid for every network. We thus eliminate the need to use experiments to adapt our definitions and results to each particular network. Finally, the fact that our functions consist of only two linear portions, shorten the execution time of the optimisation.

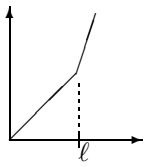



Figure 6.2: The link cost function $C^{\ell,\lambda}$.

### 6.3.3 The Result

To formulate our result we need to introduce some notation. Let $y = (y_{ij}^t)$ be a point of $P_{12}$ or $P_1$, and suppose given real numbers $\lambda > 1$ and $\ell > 0$. We define the link cost function (illustrated in Fig. 6.2)

$$C^{\ell,\lambda}(U) = \begin{cases} U & \text{if } U \leq \ell \\ \lambda\ U + (1-\lambda)\ \ell & \text{if } U \geq \ell \end{cases} .$$

We use this function in the definition of the following objective function:

$$f^{\ell,\lambda}(y) = \sum_{(i,j)\in E} k_{ij}\ C^{\ell,\lambda}(u(y,i,j))$$

We also need to define the following constants:

$$v = \min\{f(y) \,|\, y \in P_{12}\} \qquad \text{and} \qquad V = \max\{f(y) \,|\, y \in P_{12}\}$$

Notice that $v > 0$ since $D(s,t) > 0$, and $V < \infty$ since the network is finite and we are enforcing the (finite) capacity conditions. At a more practical level, $v$ can be computed by simply feeding the linear problem $\min\{f(y) \,|\, y \in P_{12}\}$ into CPLEX and solving it. Then, to compute $V$, one changes the same linear problem to a max problem (by replacing "min" by "max") and solves it.

Finally, let $\delta > 0$ denote the minimum capacity of the edges of positive capacity. We can now state the following theorem whose proof is given in a technical report [9]:

**Theorem 1**. *Let $\ell$, $\epsilon$ be real numbers satisfying $0 < \ell < 1$ and $0 < \epsilon < 1 - \ell$. Suppose that $y \in P_1$ is $\ell$-balanced, and let $\lambda > 1 + \frac{V^2}{v\delta\epsilon}$. Then any solution $x$ of $MCF_1$ with objective function $f^{\ell,\lambda}$ is $(\ell+\epsilon)$-balanced. Moreover, $x$ is more efficient than any other $(\ell+\epsilon)$-balanced point of $P_1$.*

Observe that, since $\ell < 1$ and $y \in P_1$ is $\ell$-balanced, we can use $MCF_1$ instead of $MCF_{12}$. Informally, the theorem says that if there are $\ell$-balanced

solutions, then $f^{\ell,\lambda}$ will find one. The number $\epsilon > 0$ is a technicality needed in the proof. Notice that it can be chosen arbitrarily small.

Theorem 1 can be used as follows. Given a target utilisation $\ell$, say $\ell = 0.7$, compute $\frac{V^2}{v\delta\epsilon}$, choose a $\lambda$ as in Theorem 1, and choose $\epsilon > 0$, say $\epsilon = 0.01$. Finally, compute a solution, say $x$, of $MCF_1$ with objective function $f^{\ell,\lambda}$. Then there are two exclusive possibilities: either $x$ is $0.71$-balanced or there is no such solution. In the last case, $x$ can be thought of as a "best effort" solution since we have penalised all utilisation above $0.7$ (which forces traffic using edges to more than 70 % of capacity to try to balance) but no $0.71$-balanced solution exists. At this point we can either accept this best effort solution or iterate, this time setting the balance target to, say, $0.85$, etc. After a few iterations we arrive at a solution which is "sufficiently" balanced or we know that there is no solution that is $\ell$-balanced for the current value of $\ell$ which, we may decide, is so close to $1$ that it is not worthwhile to continue iterating.

### 6.3.4 A Generalisation

Theorem 1 has a useful generalisation that can be described as follows. Partition the set of edges $E$ into a family $(E_i)$ of subsets, and choose a target utilisation $\ell_i$ for each $E_i$. The generalised theorem says that for small $\epsilon > 0$ we can define a function corresponding to $f^{\ell,\lambda}$ in Theorem 1, such that solving $MCF_1$ with this objective function will result in efficient solutions that are $(\ell_i + \epsilon)$-balanced on $E_i$ provided, of course, that such solutions exist. The generalised theorem is more flexible in that it allows us to seek solutions with different utilisation in different parts of the network.

### 6.3.5 Quantitative Results

We have used CPLEX 7.1[1] on a Pentium laptop to conduct numerical experiments with a graph representing a simplified version of a real projected network. The graph has approximately 200 nodes and 720 directed edges. If we had modelled MCF with source-destination pairs as commodities, the linear problem corresponding to $MCF_{12}$ would consist of some 8 million equations and 30 million variables. Modelling commodities as traffic to a node, $MCF_{12}$ contains, in contrast, "only" about 40 000 constraints and 140 000 variables. Solving $MCF_1$ with objective function $f^{\ell,\lambda}$ takes approximately 35 seconds.

Solving the same problem with the objective function considered by Fortz and Thorup [2, 3] takes approximately 65 seconds. Our experiments suggest

[1] ILOG CPLEX 7.1 http://www.ilog.com

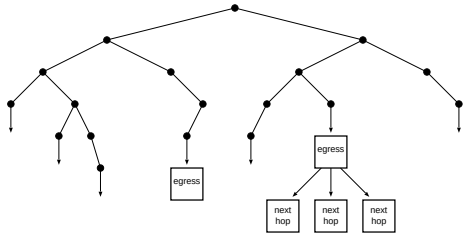



Figure 6.3: Address lookup data structure for multiple path forwarding.

that this function picks solutions that minimise balance. In contrast, with $f^{\ell,\lambda}$ we can choose any desired level of balance (above the minimum, of course).

## 6.4 Multi-Path Forwarding

By modelling the routing problem as "all traffic to $t$", as described in the previous section, we get an output from the optimisation that is well suited for packet forwarding in the routers. The result from the optimisation, the $y_{ij}^t$ values, tells how packets at a certain node ($i$) to a certain egress node ($t$) in the network should be divided between the set of next hops ($j$). We thus need a forwarding mechanism that can distinguish packets destined for a certain egress, and that can forward along multiple paths.

To enable forwarding along multiple paths, we introduce one more step in the usual forwarding process. An egress data structure is inserted in the address lookup tree just above the next hop data structure as illustrated in Fig. 6.3. A longest prefix match is done in the same manner as in a standard forwarding table, except that it results in the destination egress node. The egress data structure stores references to the set of next hops to which traffic for that egress should be forwarded, as well as the desired ratios (the $y_{ij}^t$ for all $j$s) between the next hops.

In order to populate the forwarding tables a mapping has to be created between destination addresses and egress nodes. The needed information is the same as a regular intra-domain routing protocol needs, and is obtained in much the same way. For destinations in networks run by other operators (i.e., in other routing domains), the mapping is obtained from the BGP routing protocol. For intra-domain destinations, the destination prefix is directly connected to the egress node.

Mechanisms for distributing traffic between multiple links have been thoroughly evaluated by Cao et al. [10]. We propose to use a table based hashing mechanism with adaptation, because it can distribute the load according to unequal ratios, is simple to compute, and adapts to the properties of the actual traffic.

Similar mechanisms already exist in commercial routers in order to handle the equal cost multi-path extension to OSPF and similar protocols.

## 6.5 Related Work

With the prospect of better utilising available network resources and optimising traffic performance, a lot of research activity is currently going on in the area of traffic engineering. The general principles and requirements for traffic engineering are described in the RFC 3272 [1] produced by the IETF Internet Traffic Engineering working group. The requirements for traffic engineering over MPLS are described in RFC 2702 [5].

Several researchers use multi-commodity flow models in the context of traffic engineering. Fortz and Thorup [2, 3] use a local search heuristics for optimising the weight setting in OSPF. They use the result of multi-commodity flow optimisation as a benchmark to see how close to optimal the OSPF routing can get using different sets of weights. Mitra and Ramakrishnan [11] describes techniques for optimisation subject to QoS constraints in MPLS-supported IP networks. Poppe et al. [8] investigate models with different objectives for calculating explicit routes for MPLS traffic trunks. Multi-commodity flow and network flow models in general have numerous application areas. A comprehensive introduction to network flows can be found in Ahuja et al. [7].

A somewhat controversial assumption when using multi-commodity flow optimisation is that an estimate of the demand matrix is available. The problem of deriving the demand matrix for operational IP networks is considered by Feldmann et al. [12]. The demand matrix only describes the current traffic situation but, for an optimisation to work well, it must also be a good prediction of the near future. Current research in traffic analysis by Bhattacharyya et al. [6] and Feldmann et al. [12] indicate that sufficient long term flow stability exists on backbone links in timescales of minutes and hours and in manageable aggregation levels to make optimisation feasible.

## 6.6 Conclusions

We have taken the first steps to introduce flow optimisation as a routing mechanism for an intra-domain routing protocol. We have presented a routing algorithm based on multi-commodity flow optimisation which we claim is computationally tractable for on-line routing decisions and also only require a small modification to the legacy packet forwarding mechanism. More work is however needed on other components in order to design and implement a complete routing protocol using our algorithm.

The key issue, and our main contribution, is the mathematical modelling of commodities. Traffic destined for a certain egress node is aggregated into a single commodity. This results in computational requirements an order of magnitude smaller than in the traditional models where the problem is modelled with one commodity for each flow from one ingress to one egress node.

Multi-path forwarding of the aggregates produced by the optimiser is then handled by a hash based forwarding mechanism very similar to what is needed for OSPF with ECMP.

Another contribution is the design of a generic objective function for the optimisation which allows the network operator to choose a desired limit on link utilisation. The optimisation mechanism then computes a most efficient solution given this requirement, when possible, and produces a best effort solution in other cases. The process can be iterated with, e.g., binary search to find a feasible level of load balance for a given network load.

# Bibliography

[1] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of Internet traffic engineering. Internet RFC 3272, May 2002.

[2] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000*, pages 519–528, Israel, March 2000.

[3] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, May 2002.

[4] D. Yuan. *Optimization Models and Methods for Communication Network Design and Routing*. PhD thesis, Linköpings Universitet, 2001.

[5] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. Internet RFC 2702, September 1999.

[6] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft. Pop-level and access-link-level traffic dynamics in a tier-1 PoP. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, San Francisco, USA, November 2001.

[7] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows*. Prentice-Hall, 1993.

[8] F. Poppe, S. van den Bosch, P. de la Vallée-Poussin, H. van Hove, H. de Neve, and G. Petit. Choosing the objectives for traffic engineering in IP backbone networks based on Quality-of-Service requirements. In *Proceedings of First COST 263 International Workshop, QofIS*, pages 129–140, Berlin, Germany, September 2000.

[9] J. Alonso, H. Abrahamsson, B. Ahlgren, A. Andersson, and P. Kreuger. Objective functions for balance in traffic engineering. Technical Report T2002:05, SICS – Swedish Institute of Computer Science, May 2002.

[10] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for internet load balancing. In *Proceedings of IEEE INFOCOM 2000*, Israel, March 2000.

[11] D. Mitra and K. G. Ramakrishnan. A Case Study of Multiservice, Multipriority Traffic Engineering Design for Data Networks. In *Proceedings of Globecom'99*, Brazil, 1999.

[12] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. In *Proceedings of ACM SIGCOMM'00*, Stockholm, Sweden, August 2000.

# Chapter 7

# Paper B: Robust Traffic Engineering using L-balanced Weight-Settings in OSPF/IS-IS

Henrik Abrahamsson and Mats Björkman. In:*Sixth International Conference on Broadband Communications, Networks, and Systems (BROADNETS 2009)*, September 2009, Madrid, Spain.

**Abstract**

Internet traffic volumes continue to grow at a great rate, now pushed by video and TV distribution in the networks. This brings up the need for traffic engineering mechanisms to better control the traffic. The objective of traffic engineering is to avoid congestion in the network and make good use of available resources by controlling and optimising the routing function. The challenge for traffic engineering in IP networks is to cope with the dynamics of Internet traffic demands. Today, the main alternative for intra-domain traffic engineering in IP networks is to use different methods for setting the weights in the routing protocols OSPF and IS-IS.

In this paper we revisit the weight setting approach to traffic engineering but with focus on robustness. We propose *l*-balanced weight settings that route the traffic on the shortest paths possible but make sure that no link is utilised to more than a given level *l*. This gives efficient routing of traffic and controlled spare capacity to handle unpredictable changes in traffic. We present a heuristic search method for finding *l*-balanced weight settings and show that it works well in real network scenarios.

## 7.1 Introduction

Internet traffic volumes continue to grow at a great rate, now pushed on by video and TV distribution in the networks. Increasing traffic volumes necessitate upgrades of network equipment and new investments for operators, and keep up-to-date the question of over-dimensioning network capacity versus using traffic engineering mechanisms for better handling the traffic. In addition, as new bandwidth demanding and also delay and loss sensitive services are introduced, it is even more important for the operator to manage the traffic situation in the network.

The main challenge for traffic engineering is to cope with the dynamics of traffic demands and topology. How to best model and describe aggregated Internet traffic is still an open area of research. On short timescales up to seconds the traffic is very bursty and on long timescales there are often predictable daily and weekly cycles. In between there can be unpredictable changes and shifts in traffic demand, for instance due to hotspots and flash crowds, or because a link goes down, there are changes in the inter-domain BGP routing, or because traffic in an overlay is re-directed. For future networks more variability in traffic demands is also expected due to mobility of nodes and networks and more dynamic on-demand service level agreements (SLA:s).

The traffic variability means that, even if we could measure the current traffic situation exactly, it would not always correctly predict the near future traffic situation and this needs to be taken into account when doing traffic engineering. Network operators often handle this by relying on simple well-tried techniques (like OSPF and IS-IS routing), over-dimensioning of network capacity, and simple rules of thumb (i.e upgrade the link capacity when mean utilisation reaches 70-80%) rather than introducing complex traffic engineering techniques.

In this paper we take this need for spare capacity and simple rules of thumb as our starting point. We revisit the approach of using weight settings in OSPF/IS-IS for traffic engineering but now with focus on robustness. We propose weight settings that we call *l-balanced* where the operator, by setting the parameter $l$ (to say 80%), control the maximum utilisation level in the network and how much spare capacity is needed to handle unpredictable traffic changes. With an $l$-balanced routing the traffic takes the shortest paths possible but makes sure that no link is utilised to more than a given level $l$, if possible.

The main contributions in this paper are:

- We propose $l$-balanced weight settings in OSPF/IS-IS for robust traffic engineering.

- We present a heuristic search method for finding *l*-balanced weight settings and show that it works well in real network scenarios.
- We evaluate *l*-balanced routing and compare it with other proposed traffic engineering objectives for several real network topologies and traffic data sets.

If traffic levels continue to grow then of course network capacity needs to be added at some point. But traffic engineering with *l*-balanced routing can extend the upgrade cycle and postpone the investment, or be applied to better use the existing resources in the network until the highly utilised links have been upgraded.

The paper is organized as follows. Section 7.2 gives a short introduction to traffic engineering in IP networks and Section 7.3 discusses related work. We then present the *l*-balanced cost function in Section 7.4 and describe the search heuristic used for finding *l*-balanced weight settings. In Section 7.5 we evaluate the proposed methods. We show that the search heuristic works well for finding *l*-balanced weight settings in real traffic scenarios. Further, we compare the robustness of different weight-setting methods and investigate what happens to link utilisations in the network if a traffic demand suddenly increases. Finally, in Section 7.6 we make some concluding remarks about our findings.

## 7.2 Traffic Engineering in IP networks

The objective of traffic engineering is to avoid congestion in the network and to make better use of available network resources by adapting the routing to the current traffic situation. The traffic demands in a network changes over time and for network operators it is important to tune the network in order to accommodate more traffic and meet service level agreements (SLAs) made with their customers. This means that a network operator can not rely only on long-term network planning and dimensioning that are done when the network is first built. Robust traffic engineering mechanisms are needed that can adapt to changes in traffic demand and distribute traffic to benefit from available resources.

The first step in the traffic engineering process is to collect the necessary information about network topology and the current traffic situation. Most traffic engineering methods need as input a traffic matrix describing the demand between each pair of nodes in the network. The traffic matrix is then used as

input to the routing optimization step, and the optimized parameters are finally used to update the current routing.

Today, the main alternative for intra-domain traffic engineering in IP networks is to use different methods for setting the weights (and so decide upon the shortest paths) in the routing protocols OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System). These are both link-state protocols and the routing decisions are based on link costs and a shortest (least-cost) path calculation. With the equal-cost multi-path (ECMP) extension to the routing protocols the traffic can also be distributed over several paths that have the same cost. These routing protocols were designed to be simple and robust rather than to optimize the resource usage. They do not by themselves consider network utilisation and do not always make good use of network resources. The traffic is routed on the shortest path through the network even if the shortest path is overloaded and there exist alternative paths. It is up to the operator to find a set of link costs (weights) that is best suited for the current traffic situation and that avoids congestion in the network.

The general problem of finding the best way to route traffic through a network can be mathematically formulated as a multi-commodity flow (MCF) optimization problem (see, e.g., [1, 2, 3]). The network is then modeled as a graph. The problem consists of routing the traffic, given by a demand matrix, in the graph with given link capacities while minimizing a cost function. With no limitations on how the traffic flows can be divided over the network links the MCF optimal routing problem can be formulated and efficiently solved as a linear program. Introducing integer weights and ECMP shortest paths constraints, where the traffic no longer can be split arbitrarily, makes the problem computationally much harder. For reasonably sized networks one usually has to rely on search heuristics for determining the set of weights, rather than calculating the optimal weights.

## 7.3 Related work

Traffic engineering by finding a suitable set of weights in OSPF/IS-IS is a well studied area of research and it is described in recent textbooks in the area [3, 4]. When we now revisit the weight setting approach to traffic engineering we are most inspired by the pioneering works by Fortz and Thorup [2, 5] and Ramakrishnan and Rodrigues [6], in that we use a piece-wise linear cost function and search heuristics to find suitable weight settings.

Several studies [2, 7, 8, 9] have shown that even though we limit the routing

of traffic to what can be achieved with weight-based ECMP shortest paths, and not necessarily the optimal weights but those found by search heuristics, it often comes close to the optimal routing for real network scenarios. How the traffic is distributed in the network very much depends on the objectives, usually expressed as a cost function, in the optimisation. An often proposed objective function is described by Fortz and Thorup [2] (and we will refer to it as the FT cost function further on). Here the sum of the cost over all links is considered and a piece-wise linear increasing cost function is applied to the flow on each link. The basic idea is that it should be cheap to use a link with small utilization while using a link that approaches 100% utilisation should be heavily penalized. The *l*-balanced cost function [1, 10] used in this paper is similar in that it uses a piecewise linear cost function to obtain desireable solutions. Additionaly, our cost function gives the operator the opportunity to set the maximum wanted link utilisation. Cost functions for traffic engineering is further investigated by Balon *et.al* [11].

This paper add to existing work on weight settings by focusing on robustness and the objective of achieving a controlled spare capacity for handling unpredictable traffic shifts. For robust traffic engineering much of the focus is on handling multiple traffic matrices and traffic scenarios [5, 12, 13, 14, 15, 16] and handling the trade-off between optimising for the common case or for the worst case. There are also several works on finding weight settings that are robust to link failures [17, 18, 19].

Xu *et.al* [20] describe a method to jointly solve the flow optimization and the link-weight approximation using a single formulation resulting in a more efficient computation. Their method can also direct traffic over non-shortest paths with arbitrary percentages. Their results should also be directly applicable to our problem of providing robustness to changes, by just substituting their piece-wise linear cost function with our cost function. In a continuation on that work Xu *et.al* [21] propose a new link-state routing protocol. The protocol splits traffic over multiple paths with an exponential penalty on longer paths and achieves optimal traffic engineering while retaining the simplicity of hop-by-hop forwarding.

## 7.4 L-balanced solutions

### 7.4.1 Optimal l-balanced routing

A routing is said to be *l-balanced* if the utilisation is less than or equal to *l* on every link in the network. For instance a solution is (0.7)-balanced if it never uses any link to more than 70% of its capacity.

The *l*-balanced cost function, its theoretical foundation, and use in MCF optimisation is described in [1, 10]. The idea is to use a simple piece-wise linear cost function as shown in Figure 7.1 and apply it to the utilisation of each link in the network. The cost function consists of two linear portions where the slope of the second line segment should be large enough to penalise utilisation above *l* and balance traffic over longer paths.

The work in [1, 10] present a formula to calculate the cost function, for a given network topology and traffic situation, that guarantees to find a *l*-balanced optimal routing (provided, of course, that such solutions exist) that takes the shortest paths possible and makes sure that no link is utilised to more than *l*.

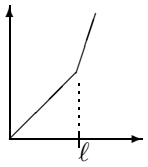



Figure 7.1: The link cost function.

### 7.4.2 Search for l-balanced weight settings

To apply the *l*-balanced routing in real OSPF/IS-IS networks we need to find *l*-balanced weight settings. For weight settings we dont have the guarantee to find an *l*-balanced routing in the same way as described for optimal routing above. But we want to use the *l*-balanced cost function to find weights settings that achieve the same effect of taking the shortest paths possible while routing the traffic so that no link is utilised to more than a given level *l*.

The problem of finding the optimal weight setting is NP-hard [2, 3]; and so the optimal weights are often too computationally hard and time consuming to calculate for real networks and traffic scenarios. Instead we use a problem specific local search heuristic to determine the set of weights. An overview of

local search methods can be found in [22]. Our search method can be placed under the Tabu search meta-heuristic in that we allow cost-increasing solutions to direct the search away from local minima, and use a tabu list to prevent from looping back to old solutions. A solution is a vector $w = \{w_1, .., w_n\}$ of weights, with one weight per directed link in the network. We have a solution space $W$ where each weight can take integer values between $1$ and $65535$. We generate a neighboring solution $i \in N(w)$ by increasing one weight in the current solution $w$ to divert traffic from the most utilised link $(s, t)$ or change weights to create paths with the same cost to get ECMP routing of traffic over several links from $s$. We use a $l$-balanced cost function (as described in the previous section) calculated for the given topology, traffic matrix and required utilisation level $l$. The cost $f(w)$ for a given weight vector is determined by calculating the shortest paths routing with these weights using Dijkstra's algorithm, adding the traffic matrix, and applying the cost function to the resulting link loads.

The starting point is to set all weights to the same value, for instance $w_i = 10$. The search terminates either when we find a solution with utilisation under the threshold $l$ or it stops after a fixed number of iterations.

At the core of our search method is a simple descent search [22] where we:

1. choose an initial weight vector $i \in W$
2. find the neighbor $j \in N(i)$ with lowest cost i.e. $f(j) <= f(k)$ for any $k \in N(i)$.
3. If $f(j) >= f(i)$ then stop. Else set $i = j$ and go to step 2.

This type of search may stop at a local minimum. We therefore allow the search to continue by doing new descents starting from weight sets with higher cost. We use information that becomes available during the search to build a candidate list of weight sets that are used as starting points, and a tabu list of weight sets are used to avoid cycling.

We start by setting all weights to the same value. This gives the shortest paths in number of hops which probably is a good starting point for most real networks; if the link capacities are uniform and the network was built with OSPF/IS-IS routing in mind. Given the network topology, traffic matrix and initial weights, we calculate the ECMP shortest paths, add the traffic matrix, and find the most loaded link $(s, t)$ in the network. If the utilisation is less than *l* then we are done. We have a routing that takes the shortest paths possible and makes sure that no link is utilised to more than the limit *l*. If the link is

utilised to more than *l* we start searching for a better weight setting using two strategies:

- the first search strategy is to increase the weight on the overloaded link in controlled steps so to divert more and more demands (or part of demands) from the link. See details in 7.4.3.
- the second search strategy is to find weights to get ECMP routing from $s$ for the demands over $(s,t)$, and so balance the traffic over the outgoing links from $s$. See details in 7.4.4.

In each iteration of a descent we have a number of neighbor weight settings that we evaluate (one for each weight step and ECMP set described above). If a neighbor weight setting gives a lower cost than the current best (in this iteration) it is saved and used as the starting point in the next iteration. If a candidate weight setting gives a routing with a higher cost than the current best but with a different link than $(s,t)$ as most utilised, then that weight-setting is saved in the candidate list and used as a starting point for another descent search later on.

### 7.4.3 How to determine weight increments for a link?

If a link $(s,t)$ is over-utilised we want to increase the weight on the link in controlled steps so to divert more and more traffic demands from the link.

To decide the steps in which to increase the weight on $(s,t)$ we first determine the current total weight-cost for each demand routed over $(s,t)$. We then temporarily take away the link $(s,t)$ from our representation of the topology and calculate a new shortest-path routing. For all demands that before were routed over $(s,t)$ we then check how much the weight cost have increased and use this for determining the steps with which to increase the weight on $(s,t)$.

In the example in Figure 7.2, we assume that the two demands $D(1,2)$ and $D(4,2)$ overload the link $(1,2)$. We thus want to divert traffic from the link $(1,2)$ by increasing the weight $w(1,2)$.

We start by determining the increase steps in which to increase the weight $w(1,2)$:

The total weight costs for $D(1,2)$ and $D(4,2)$ are 10 and 40, respectively. If we take away the link $(1,2)$, we get total weight costs of 30 and 50, an increase by 20 and 10 units respectively. From this we decide on the increase steps 10, 15 (mid-point between 10 and 20), 20 and 21 units. We add this to the original $w(1,2) = 10$ and get the candidate weights $w(1,2)$= 20, 25, 30 and 31 to evaluate.

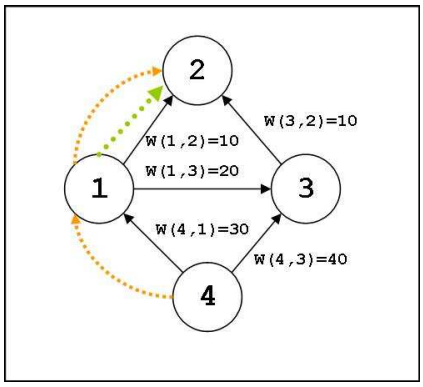



Figure 7.2: Example with an overloaded link (1,2) where traffic can be diverted to other paths by increasing the weight on (1,2) in controlled steps w(1,2)= 20, 25, 30 and 31. With the first increment w(1,2)=20 we divert half of demand D(4,2) by ECMP. The next increment w(1,2)=25 diverts all of D(4,2), and with w(1,2)=30 we route also half of D(1,2) on another path. Finally, w(1,2)=31 diverts all traffic from (1,2).

With the first increment $w(1,2) = 20$ we divert half of demand $D(4,2)$ by ECMP while the other half of $D(4,2)$ and all of demand $D(1,2)$ is still routed on $(1,2)$. The next increment $w(1,2) = 25$ diverts all of $D(4,2)$ but keeps all of $D(1,2)$. With $w(1,2) = 30$ we also route half of $D(1,2)$ on another path and with $w(1,2) = 31$ we divert all traffic from $(1,2)$.

### 7.4.4 How to determine ECMP weight settings?

If we have a weight set that results in an overloaded link $(s,t)$ then we want to also evaluate neighbor weight settings where we split traffic demands evenly over the outgoing links from $s$ using ECMP. In order to split a traffic demand ECMP the total weight for each path from $s$ to the demand destination $d$ need to be the same.

Consider, as in Figure 7.3, a node $s$, the next hops $t_i$, and the shortest path $P_i$ from each $t_i$ to the destination $d$. Also consider the corresponding weights $w(s,t_i)$ and total weight cost $w(P_i)$ for a path $P_i$ from $t_i$ to $d$. One way to achieve ECMP weights is to adjust the weights $w(s,t_i)$ on the outgoing links

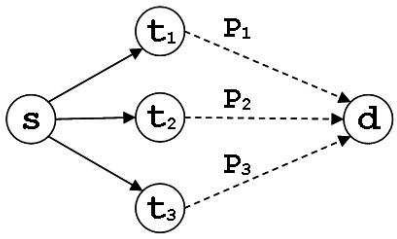



Figure 7.3: Determining ECMP weights

from $s$ such that:

$$w(s, t_i) = 1 + max_{j=1,..,n}\{w(P_j)\} - w(P_i)$$

This gives the same total cost for each path from s to d.

A possible extension to this is to not always spread the traffic over all possible links but also evaluate different subsets of ECMP weights setting with varying number of outgoing links from $s$.

### 7.4.5 Increment weight on a less utilised link in a path

With high traffic load in the network, link weights can become sensitive to change after some iterations in the search. For instance if we on an overloaded link already have adjusted the weight to split a large demand with ECMP then we can not easily increase the link weight to divert yet another flow without disturbing the existing load balancing.

In order to divert traffic demands to other paths but without disturbing existing splits on the most utilised link we extend the neighborhood in the search. We evaluate weight sets where we instead of changing the weight on the overloaded link $(s, t)$ increment the link weight some step away closer to the demand destination. In the example in Figure 7.4, assume that the link (1,3) is overloaded. With our search (as described in 7.4.3) we would in this example evaluate a weight setting where the demand $D(1, 4)$ is diverted to the path 1-5-4 by increasing the weight $w(1, 3)$ to 21. But increasing the weight $w(1, 3)$ will also send all of $D(1, 2)$ on the link $(1, 2)$, possibly creating overload on that link and a higher cost solution.

With the extended neighborhood we also evaluate alternative weight settings where we increase the weight on other links in the path (not only on the

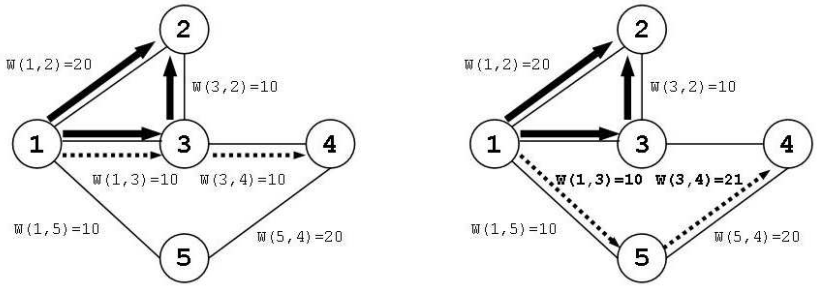



Figure 7.4: Example with an overloaded link (1,3). With an extended neighborhood in the search the demand D(1,4) can be diverted by increasing the weight w(3,4) instead of w(1,3), and avoid disturbing the other flows on the overloaded link (1,3).

most utilised link). In this example for demand $D(1,4)$ we increase the weight $w(3,4)$ which diverts the demand $D(1,4)$ from the overloaded link $(1,3)$ while keeping the needed ECMP split of demand $D(1,2)$.

### 7.4.6 Comments on the search method

As described above several different techniques are needed to get an efficient search method to find *l*-balanced solutions.

When designing and implementing our search method we were in part inspired by the works of Ramakrishnan and Rodrigues [6] and Fortz and Thorup [2]. From the first we borrowed the idea of temporarily taking away the overloaded link from the representation of the topology, and calculate a new shortest-path routing, to find the weight increments for the link. But apart from this idea our approaches are different. Fortz and Thorup [2] use a Tabu local search heuristic to find appropriate link weights, and from here we also borrowed the idea on how to find ECMP weight-settings over many links.

But, for efficiency, we wanted a more problem-specific search heuristic rather than a generic Tabu search. Instead of searching at random, we start with the shortest paths possible and directly look at the most loaded link. If the utilisation is less than *l*, then we are done and no search is needed. If the link is utilised to more than *l*, then we start to divert traffic from there.

The higher the traffic level the more difficult it is to find a weight setting, that not only balances the traffic, but actually keeps it under a specified level *l*.

We combined the existing techniques described above: weight increments and ECMP traffic splits at the most utilised link. But with our direct approach and at high traffic loads, it turned out not to be enough to find *l*-balanced routings.

Therefore, we also added our ideas with candidate lists and extended neighborhoods. For the candidate list, we choose weight settings with a higher cost but where the overload has moved to another link, in order to diverse the search. And for extended neighborhoods, we increment the weight on a less utilised link in a path in order to not disturb the weight composition in sensitive, highly loaded areas.

## 7.5 Evaluation

### 7.5.1 Method

In order to evaluate the *l*-balanced routing and our search method for finding *l*-balanced weights we use real network topologies and traffic matrix data that we scale up to get high loads in the networks. First in Section 7.5.2 we evaluate that the search method works well for finding *l*-balanced weight setting in these scenarios and compare the resulting network loads with optimal *l*-balanced routing and routing with other traffic engineering objectives. The main objective of *l*-balanced routing is to give a controlled amount of spare capacity to handle traffic changes. In Section 7.5.3 we investigate how different weight settings handle hotspots where one traffic matrix entry increases.

For the evaluation we here use two different data sets that include network topologies and traffic matrix data from the Geant network [23], and from the American sub-network of a global IP network.

- Network I: the Geant network with 23 nodes, 74 links and 506 demands.
- Network II: the American network with 24 nodes, 110 links and 552 demands.

The details of the global IP-network, the subnetwork topologies and traffic demands, are described in [24]. For the Geant network we set all link capacities to 10 Gb and scaled up the traffic data to create high loads in the network.

### 7.5.2 Static scenario: Evaluating the search method

The evaluation shows that the *l*-balanced objective and our search method for finding l-balanced weight settings work well. Figure 7.5 shows comparisons

Table 7.1: Performance of the search heuristic for network I

| load- | Min. hop routing | | L-balanced routing (L=0.8) | | | |
|---|---|---|---|---|---|---|
| level | links >L | max util | time | max util | descents | sets |
| 7 | 0 | 0.751 | 0.1 s | 0.751 | 0 | 0 |
| 8 | 2 | 0.858 | 0.3 s | 0.784 | 1 | 63 |
| 9 | 3 | 0.965 | 0.2 s | 0.797 | 1 | 51 |
| 10 | 3 | 1.072 | 0.4 s | 0.780 | 1 | 82 |
| 11 | 4 | 1.179 | 0.4 s | 0.795 | 1 | 91 |
| 12 | 5 | 1.287 | 0.4 s | 0.794 | 1 | 123 |
| 13 | 6 | 1.394 | 56.8 s | 0.790 | 262 | 21451 |

Table 7.2: Performance of the search heuristic for network II

| load- | Min. hop routing | | L-balanced routing (L=0.8) | | | |
|---|---|---|---|---|---|---|
| level | links >L | max util | time | max util | descents | sets |
| 7 | 0 | 0.728 | 0.4 s | 0.728 | 0 | 0 |
| 8 | 1 | 0.832 | 0.4 s | 0.732 | 1 | 41 |
| 9 | 2 | 0.936 | 0.6 s | 0.766 | 1 | 95 |
| 10 | 5 | 1.040 | 2.2 s | 0.732 | 7 | 671 |
| 11 | 5 | 1.144 | 477.5 s | 0.801 | 4390 | 200037 |

of optimal and weight-based *l*-balanced routing (with *l*=80%) for increasing levels of traffic demand in the Geant network (Network I) and the American network (Network II). The *l*-balanced routing sends the traffic on the shortest paths as long as the utilisation is low in the network. The shape of the curves shows that when we scale up the traffic demand the *l*-balanced method tries to keep the utilisation under *l=0.8*. The figures also show that the weight-based routing is close to the optimal routing which validates that our search method for setting the weights works well. Note that optimal routing minimises the total cost when the *l*-balanced cost function is applied to the utilisation of each link in the network. The utilisation for an individual link (and so the maximum link utilisation) can be higher in the optimal solution if it finds a shorter path that still keeps the utilisation below *l*.

Tables 7.1 and 7.2 describe the performance of our search method and show that our search heuristic is fast. The left-hand side of the tables describes the load situation in the networks. The increasing load levels (shown in the first column) come from multiplying each entry in the traffic matrix with a

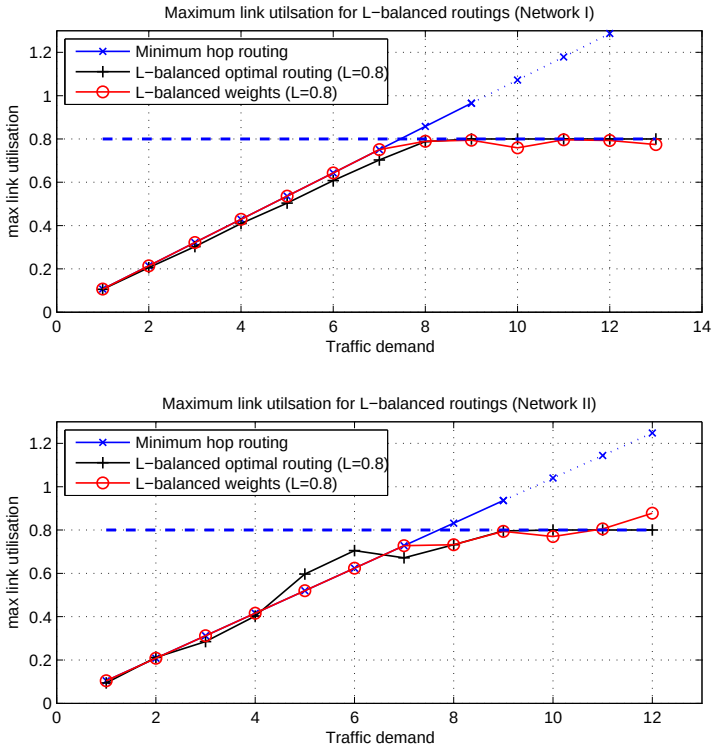



Figure 7.5: Comparison of maximum link utilisations for optimal and weight-based L-balanced routing for different scaled traffic demands in the Geant network (top) and the American network (bottom). The utilisation is kept under the chosen limit $l$ and the weights found by the search heuristic gives a routing close to optimal.

higher and higher constant value. For both networks it holds that, up to level 7, no search is needed since the start weights (all set to 10) and the resulting minimum-hop routing give a maximum link utilisation of less than $l = 0.8$.

The second and third columns show the number of links that are loaded to more the level $l = 0.8$ and the maximum link utilisation, when all weights are set to 10. This is the state from which the search start.

The right-hand side of the tables shows the performance of our search method. The first column shows how long time it takes to find an *l*-balanced solution for different levels of network load. The table also shows the number of search descents (number of new starts) and the total number of neighbor weight sets that were evaluated.

As an example, for Network I in Table 7.1, at scale 8 there are two links that are utilized to more than $l = 0.80$. The search heuristic investigate 63 different weight settings to find an *l*-balanced solution with a maximum link utilisation of 0.784. This search took only 0.3 seconds on a standard laptop with a 1.6GHz Intel Core 2 Duo CPU and 2 GB of memory.

At scale 13 there are 6 links utilized to more than $l = 0.80$ and with a maximum utilisation of 1.394. The search needs to find a weight setting that diverts traffic and simultaneously pushes down all six link utilisations under $l = 0.8$ (and without increasing any other link to more than $l$, of course). Our search heuristic evaluates 21451 weight settings and finds an *l*-balanced solution at this level in less than a minute.

Figure 7.6 shows a comparison between the *l*-balanced routing and other traffic engineering objectives. The minimum-hop routing (with all weights set to 10), where no attempt is done to adapt the weight setting to the current traffic demand, quickly leads to overload in the network when the traffic demands are increased. The *l*-balanced method sends the traffic on the shortest paths as long as the utilisation is less than the chosen value *l*=0.8. With a low utilisation of the network there is no reason to split the traffic over several paths. The FT cost function used in [2], pushes down the maximum link utilisation already at lower traffic levels. This piece-wise linear cost function consists of several segments which is reflected in the shape of the curve with plateaus where the maximum link utilisation is pushed down.

With minmax routing the objective is to minimise the maximum link utilisation in the network. This routing always balance the load over the network to keep the highest link utilisation down to a minimum. The optimal minmax routing gives a lower bound on how much it is possible to keep down the maximum link utilisation.

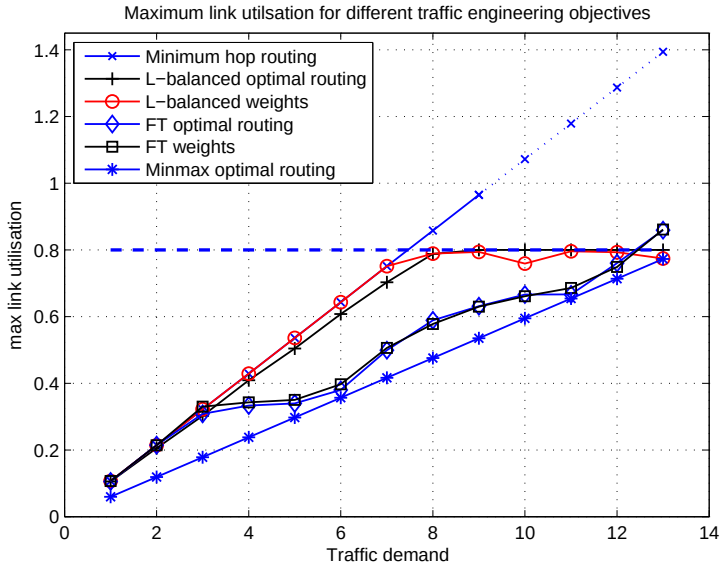



Figure 7.6: Comparison of maximum link utilisations for different traffic engineering objectives in the Geant network.

### 7.5.3 Dynamic scenario: Evaluation of robustness

The main purpose with *l*-balanced routing is to give a controlled traffic level and spare capacity to handle uncertainties and sudden changes in the traffic situation. To confirm that the *l*-balanced weight settings fulfil this, we added hotspot traffic (in a magnitude that the *l*-balanced routing should be able to handle) and investigated the resulting link utilisations. Figure 7.7 shows the maximum link utilisations for minimum hop routing, *l*-balanced and FT weight-settings under assumed hotspot traffic in the Geant network scenario.

After determining the weights and the routing for a given traffic matrix each of the 506 demands was increased one at a time by 20% of the link capacity.

The minimum hop routing, without any traffic engineering, gives link overload for all hotspot traffic at this demand level. The FT routing sometimes results in overloaded links when the hotspot traffic is added. The *l*-balanced routing (with $l$=0.8) on the other hand gives 20% spare capacity and so handle the increase for any of the demands.

## 7.6 Conclusions

In this paper we propose *l*-balanced routing with OSPF/IS-IS for robust traffic engineering. We present a heuristic search method for finding *l*-balanced weight settings and show that the search and the resulting weight settings work well in real network scenarios.

L-balanced weight settings give the operator possibility to apply simple rules of thumb for controlling the maximum link utilisation and control the amount of spare capacity needed to handle sudden traffic variations. It gives more controlled traffic levels than other cost functions and more efficient routing for low traffic loads when there is no need to spread traffic over longer paths.

Our local search method can be placed under the Tabu search meta-heuristic in that we allow cost-increasing solutions to direct the search away from local minima, and use a tabu list to prevent from looping back to old solutions. But for efficiency, rather than using a generic Tabu search, we implement a search heuristic specific for the problem of finding *l*-balanced weight settings. We start with minimum-hop routing and investigate the most loaded link. If the utilisation is less than *l*, then we are done and no search is needed. If the link is utilised to more than *l*, then we start the search from there, and we use several different weight strategies for diverting traffic to other paths.

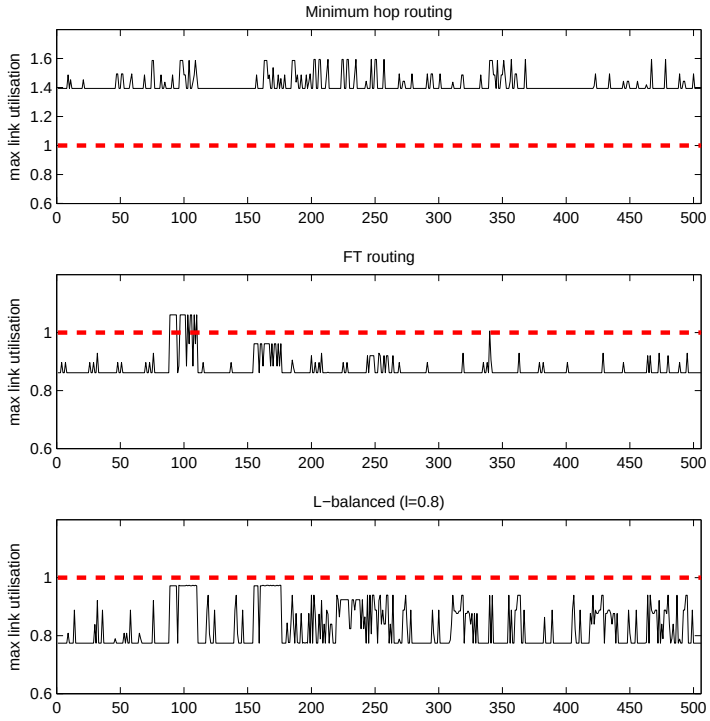



Figure 7.7: Hotspot traffic scenario in the Geant network. Comparison of maximum link utilisations for three weight setting strategies. Minimum hop routing and FT routing exceeds the link capacity while $l$-balanced routing can avoid overload.

The higher the traffic level the more difficult it is to find a weight setting, that not only balances the traffic, but actually keeps it under a specified level $l$. We combine controlled weight increments and ECMP traffic splits to divert traffic from the most utilised link. We also introduce candidate lists and extended neighborhoods. Promising weight settings that move the overload to other links are saved in the candidate list to be starting points for further search. Extended neighborhoods means that, when diverting a traffic flow from an overloaded link, we do not only try to increase the weight on the overloaded link. We also evaluate weight settings where we increment the weight on a less utilised link further down the path. This is done in order to not disturb already achieved traffic splits in highly loaded areas.

We evaluate our search heuristic in several real network scenarios and show that the search is fast and that it finds $l$-balanced weight-settings in seconds or minutes depending on the traffic level.

## Acknowledgment

The authors would like to thank Anders Gunnar for providing the traffic data used in the evaluation. We also would like to thank Adam Dunkels and Bengt Ahlgren for inspiring and helpful discussions.

# Bibliography


[1] H. Abrahamsson, J. Alonso, B. Ahlgren, A. Andersson, and P. Kreuger. A multi path routing algorithm for IP networks based on flow optimisation. In *Proceedings of the Third International Workshop on Quality of Future Internet Services (QoFIS)*, Zürich, Switzerland, October 2002.

[2] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000*, pages 519–528, Israel, March 2000.

[3] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann, 2004.

[4] J. Rexford. Route optimization in IP networks. In Mauricio G.C. Resende and Panos M. Pardalos, editors, *Handbook of Optimization in Telecommunications*. Springer Science+Business Media, 2006.

[5] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, May 2002.

[6] K.G. Ramakrishnan and M.A. Rodrigues. Optimal routing in shortest path data networks. *Lucent Bell Labs Technical Journal*, 6(1), 2001.

[7] A. Gunnar, H. Abrahamsson, and M. Söderqvist. Performance of Traffic Engineering in Operational IP-Networks: An Experimental Study. In *Proceedings of 5th IEEE International Workshop on IP Operations and Management*, Barcelona, Spain, October 2005.

[8] A. Sridharan, R. Guérin, and C. Diot. Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks. In *IEEE Infocom*, San Francisco, March 2003.

[9] David Applegate and Edith Cohen. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs. In *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[10] J. Alonso, H. Abrahamsson, B. Ahlgren, A. Andersson, and P. Kreuger. Objective functions for balance in traffic engineering. Technical Report T2002:05, SICS – Swedish Institute of Computer Science, May 2002.

[11] S. Balon, F. Skivee, and G. Leduc. How Well Do Traffic Engineering Objective Functions Meet TE Requirements? In *Proceedings of IFIP International Networking Conference*, Coimbra, Portugal, May 2006.

[12] C. Zhang, Z. Ge, J. Kurose, Y. Liu, and D. Townsley. Optimal Routing with Multiple Traffic Matrices: Tradeoffs between Average Case and Worst Case Performance. In *Proceedings of ICNP 2005*, Boston, USA, November 2005.

[13] C. Zhang, Y. Liu, W. Gong, J. Kurose, R. Moll, and D. Townsley. On Optimal Routing with Multiple Traffic Matrices . In *Proceedings of Infocom 2005*, Miami, USA, March 2005.

[14] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. Cope: Traffic engineering in dynamic networks. In *Proceedings of ACM SIGCOMM*, Pisa, Italy, September 2006.

[15] Anders Gunnar and Mikael Johansson. Robust routing under bgp reroutes. In *Proceedings of Globecom 2007*, Washington, DC, USA, 2007.

[16] M. Menth, R. Martin, and J. Charzinski. Capacity Overprovisioning for Networks with Resilience Requirements. In *Proceedings of the ACM SIGCOMM'06*, Pisa, Italy, September 2006.

[17] Antonio Nucci, Supratik Bhattacharyya, Nina Taft, and Christophe Diot. IGP Link Weight Assignment for Operational Tier-1 Backbones. *IEEE/ACM Transactions on Networking*, 15(4), August 2007.

[18] M. Menth, M. Hartmann, and R. Martin. Robust IP Link Costs for Multilayer Resilience. In *Proceedings of the 6th International IFIP Networking Conference*, Atlanta, Georgia, USA, May 2007.

[19] A. Sridharan and R. Guérin. Making IGP Routing Robust to Link Failures. In *Proceedings of the 4th International IFIP Networking Conference*, Waterloo, Ontario, Canada, May 2005.

[20] Dahai Xu, Mung Chiang, and Jennifer Rexford. DEFT: Distributed exponentially-weighted flow splitting. In *IEEE Infocom*, Anchorage, Alaska, USA, May 6–12, 2007.

[21] Dahai Xu, Mung Chiang, and Jennifer Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. In *IEEE Infocom*, Phoenix, Arizona, USA, April 2008.

[22] E. Aarts and J. K. Lenstra, editors. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[23] *The Geant network*. http://www.geant.net.

[24] A. Gunnar, M. Johansson, and T. Telkamp. Traffic Matrix Estimation on a Large IP Backbone - a Comparison on Real Data. In *Proceedings of ACM Internet Measurement Conference*, Taormina, Sicily, Italy, October 2004.

# Chapter 8

# Paper C: Simulation of IPTV caching strategies

Henrik Abrahamsson and Mats Björkman.
In *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'10)*, 11-14 July 2010, Ottawa, Canada.

**Abstract**

IPTV, where television is distributed over the Internet Protocol in a single operator network, has become popular and widespread. Many telecom and broadband companies have become TV providers and distribute TV channels using multicast over their backbone networks. IPTV also means an evolution to time-shifted television where viewers now often can choose to watch the programs at any time. However, distributing individual TV streams to each viewer requires a lot of bandwidth and is a big challenge for TV operators. In this paper we present an empirical IPTV workload model, simulate IPTV distribution with time-shift, and show that local caching can limit the bandwidth requirements significantly.

## 8.1 Introduction

IPTV, where TV channels are distributed using IP multicast, has become popular and widespread. Many telecom and broadband companies have become TV providers and distribute TV channels using multicast over their backbone network. IPTV also means an evolution to time-shifted TV where viewers can choose to watch the programs at anytime.

When distributing the TV schedule using multicast there is only one TV stream per channel, while for time-shifted TV there can be one stream per customer. Distributing individual TV streams to each viewer requires a lot of bandwidth and this is a big challenge for TV operators. The operators now therefore only gradually introduce access to more and more time-shifted TV programs, and try out different technical solutions.

TV statistics show that there is a small set of very popular programs that most people are watching. A popular prime-time program that is scheduled and distributed with multicast at a given time, will most likely also have a lot of viewer that choose to watch the program time-shifted a bit later during the evening. For an operator with many hundred thousands of TV subscribers there can be a very large number of copies of the same popular content distributed and putting load on the network.

The question we address in this work is: *to what extent can we limit the bandwidth requirements from time-shifted TV by caching the most popular programs closer to the viewers?* The answer depends on several factors including cache size, caching strategy, and the viewers' request pattern for TV programs.

Caching is a well studied technique for web content and video [1, 2, 3, 4, 5, 6] and have started to attract interest also in the context of IPTV [7, 8, 9, 10].

In order to develop and evaluate IPTV caching strategies good workload models are needed. In this paper we use an empirical IPTV workload model to simulate IPTV distribution with time-shift and investigate the benefit of introducing a local cache closer to the TV subscribers. The simulations are based on real TV schedules, and statistics about TV program popularity and viewer activity. We simulate a large number of TV viewers that, when active, request scheduled or on-demand programs and we investigate the resulting bandwidth requirements on the down link for different cache sizes and caching strategies.

The contributions of this paper are: We present an empirical IPTV workload model. We simulate a realistic scenario for IPTV distribution and compare the Least Recently Used (LRU) and Least Frequently Used (LFU) caching strategies. We show that time-shifted TV can be very capacity demanding and that considerable amounts of bandwidth can be saved by caching the most pop-

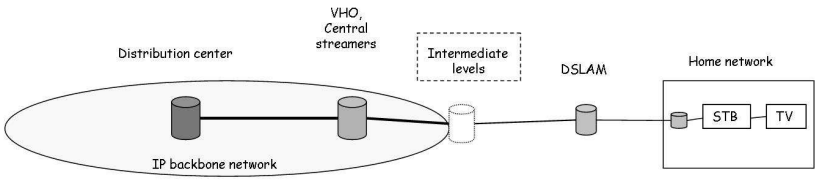



Figure 8.1: IPTV network architecture.

ular programs closer to the viewers.

The rest of the paper is structured as follows: In section 8.2 we describe IPTV distribution and time-shifted TV. In section 8.3 we present the IPTV simulator, the workload model, and the simulation scenario. The caching strategies LFU, LRU and Clairvoyant are described in section 8.4. The simulation results and the evaluation of the caching strategies are presented in section 8.5. Related work is in section 8.6, future work in section 8.7 and we conclude the paper in section 8.8.

## 8.2 IPTV and time-shifted TV

We consider IPTV distribution within in a single operator network, where the operator controls the network and how the TV content is distributed.

IPTV operators distribute traditional scheduled TV channels but also gradually introduce new TV services where the viewer can choose to watch a program later after its scheduled time. This service is called time-shifted TV (or sometimes TV on-demand or Catch-Up TV).

A typical IPTV architecture with a hierarchical tree-like network structure is illustrated in Figure 8.1. The TV content is delivered from content providers and comes into the network at a central distribution center from where it is transmitted to Video Hub Offices (VHO). A Video Hub Office has storage and video streaming equipment to serve a district or a city. Under the VHO there can be intermediate levels of storage and video servers. Different operators try and use different structures of varying complexity. The figure also shows a TV subscriber with a home network where the TV and the set-top box (STB) is connected via a residential gateway to a Digital Subscriber Line Access Multiplexer (DSLAM).

The TV channels are distributed using IP multicast from the distribution center to the set-top boxes. TV programs requested outside the schedule are streamed with unicast from the VHO (or from an intermediate server if available) to the set-top box.

## 8.3 Simulation of IPTV

### 8.3.1 Workload model

We want to investigate the load that IPTV with time-shift can put on a network and how caching can reduce the bandwidth requirements. For this we need a model of the network, a model of how TV is distributed and how TV viewers request programs and put load on the network. We need a TV schedule with channels and programs that is continuously updated, a set of on-demand programs, and a number of viewers that choose programs to watch (either live programs or time-shifted programs).

Our approach to this is to use an empirical model to simulate IPTV distribution. We have implemented a time-driven simulator that operates on the time-scale of minutes. We simulate TV distribution by stepping through real TV schedules and by using statistics about the TV programs' popularity and viewer activity.

### 8.3.2 Data set

We use a data set from traditional TV with 13 channels over 5 days from Mediamätning i Skandinavien (MMS) [11]. MMS together with Nielsen Audience Measurement [12] measure the viewing habits of the TV audience in Sweden. The measurements are done using a so called People Meter system where the viewing habits of sample households are logged using electronic meters connected to the remote control.

Our data set include 2225 TV programs from the most popular TV channels in Sweden. For each TV program we extracted the time it was scheduled, its length and the number of viewers. There are a few programs that have a very large number of viewers. The most popular program in this data set have more than 2.3 million viewers (26% of the population) while many of the programs only have a few thousand viewers. The graph in figure 8.2 shows the number of programs and their share of the total TV viewing time. The top 1% (22 of 2225 programs) most popular programs stand for 26% of the viewed TV time

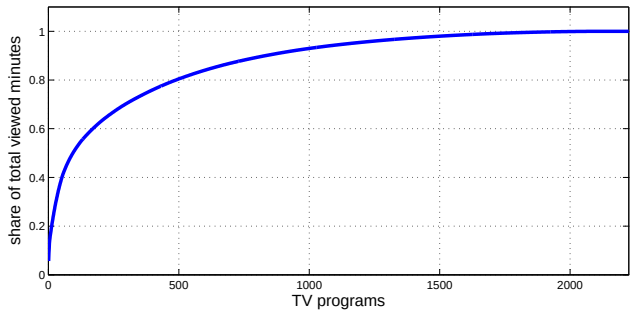



Figure 8.2: TV program popularity.

in this data set. The 10% most popular programs stand for 64% of the viewed minutes.

The data set also gives us information about the total number of viewers that are active and watch TV at any given time. Figure 8.3 shows the fraction of the viewers that are active and how it varies over the five days (Monday to Friday). There are distinct peaks in the evenings when 40-48% of the viewers are active.

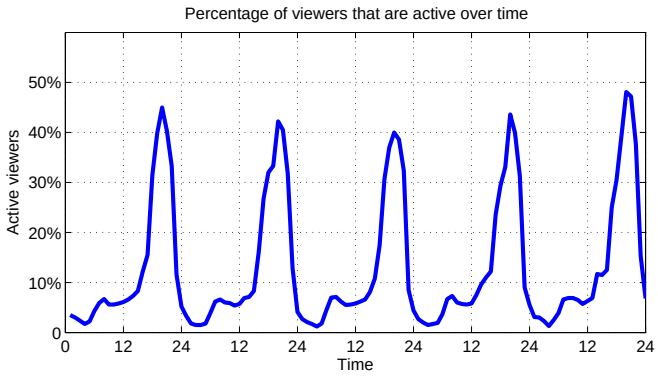



Figure 8.3: Percentage of viewers that are active.

### 8.3.3 TV programs

In the simulator we represent and step through real TV schedules. For the data set described in 8.3.2 we have a schedule with 13 channels over 5 days. So, there is a set of at most 13 ongoing channel programs available at any point in time. The data set also gives us the number of viewers of each program.

In each time step in the simulation we move forward one minute in the schedule, update the set of programs (add new programs and delete the ones that ended), and re-calculate the relative popularity of each program. The latter sets the probability that a viewer will choose to watch a particular TV program.

There is also a set of time-shifted programs that is updated in each step of the simulation. All scheduled programs goes into the set of time-shifted programs and can be requested on-demand. The first minute of a program that is scheduled (and sent out with multicast) at time $t$ is made available for time-shifted viewing at time $t + 1$. The time interval that the programs are available on-demand decides the size of the set of available programs. This a tunable parameter in the simulation. For the experiments described in section 8.5 we used a default value of 24 hours.

The popularity of the programs at a given time step in the simulation is illustrated in figures 8.4 and 8.5 for the scheduled and time-shifted programs respectively. Here no attempt is made to fit the data to well-known distributions. Instead we generate values from the empirical probability distributions using the inverse transformation method, for instance described in Jain [13]. To choose which program to watch a viewer generate a random number between 0 and 1 from a uniform distribution. For the example in figure 8.4, if the value is between 0 and 0.46 then program 1 is chosen, if the value is between 0.46 and 0.84 program 2 is chosen and so on.

### 8.3.4 TV viewers

Our TV viewers are either ON watching TV programs or OFF sleeping. In the simulator we follow the graph from the viewer statistics in figure 8.3 closely and in each time step adjust the fraction of the viewers that are active and watch TV. A viewer that is activated chooses a program to request. He chooses either to join the distribution of an ongoing scheduled program or to request one of the time-shifted programs that are available on-demand. The particular program to watch is then chosen randomly following the empirical probability distribution for the popularity of the currently available programs.

Table 8.1 shows parameter settings for the simulations we present in this

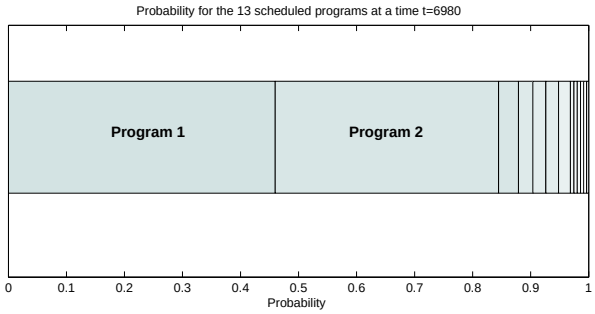



Figure 8.4: Probabilities for the 13 scheduled programs at $t = 6980$.

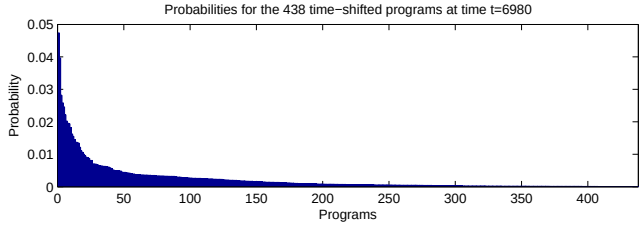



Figure 8.5: Probabilities for the 438 time-shifted programs available at $t = 6980$.

Table 8.1: Simulation parameters.

| | |
|---|---|
| Number of viewers | 1000 |
| Number of TV channels | 13 |
| Number of TV programs | 2225 |
| Programs available time-shifted | 24 hours |
| Simulated time | 7200 minutes |
| Scheduled TV/Time-shifted TV | 50/50 |
| TV stream bit rate | 2 Mbps |

paper. The share of time-shifted viewing will most likely increase with time when more programs becomes available on-demand and the viewers get used to choosing programs outside the schedule. Here we use a 50% chance that an active viewer chooses to watch a time-shifted program.

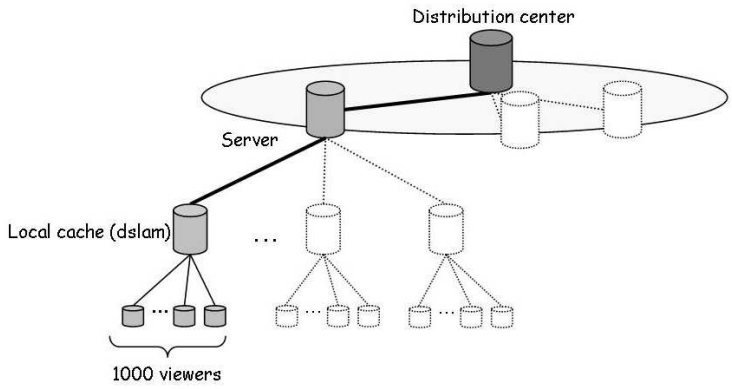



Figure 8.6: IPTV network simulation scenario. Studying the effect of introducing a local cache.

### 8.3.5 Network model and simulation scenario

In our simulator we can represent different topologies with caching at different levels in the network including in the set-top boxes. But in this work we delimit the network structure to study the effect of introducing a local cache (in the DSLAM) and the importance of cache size and caching strategy used at this node.

For this, we study one branch of an IPTV network topology (as shown in figure 8.6) with one server, a single local cache, and one thousand viewers (TV set-top boxes). We assume that all programs are distributed to the TV server and that all programs are stored there as long as they are available for time-shifted viewing. For the local cache it is different: what is stored in a local cache at a given moment in time depend on the size of the cache (which is a parameter that we investigate in the simulations), the caching strategy in use; and what programs the viewers under the cache have chosen to watch (the request pattern). We monitor the load on the link from the server to the local cache and we investigate how the bandwidth requirements varies with cache size (including the case with no caching) and caching strategy.

The bit rate of a TV stream depends on the TV channel and codec used. For simplicity we here assume that all TV streams require 2 Mbps.

If a viewer requests a scheduled program, and none of its neighbors is

watching this channel, then a multicast stream is added to the load on the link down from the server to the local cache. If someone is already watching the channel, then the new viewer joins the ongoing multicast distribution and no additional load is added to the down link. Requests for time-shifted programs first go to the local cache. If the program is not available there, it is instead transferred with unicast from the server adding 2 Mbps to the load on the down link to the local cache.

## 8.4 Caching strategies

When the cache is full, and a new program part arrives, a strategy is needed to decide what should stay in the cache and what to delete. In this work we simulate and compare three different strategies.

### 8.4.1 Least Recently Used

With the Least Recently Used (LRU) strategy we delete from the cache the program that has not been requested for the longest time.

### 8.4.2 Least Frequently Used

With Least Frequently Used (LFU) we discard the program that is requested least often. This could be done by counting the number of viewers that join the multicast distribution of a program and the number of on-demand requests. In the simulation we here use the known popularity of all programs, and delete the one with the least probability for being requested. In addition to that we only consider to delete *inactive* programs; that is programs that no one is watching at this moment in time.

### 8.4.3 Clairvoyant

In the simulation we also implement a clairvoyant strategy with the ability to look into the future and delete the program part that will not be needed for the longest time. This is done by running the simulations twice. In the first run all viewer requests are logged; and in the second run this information is used to determine which program part that will not be asked for for the longest time. The purpose of this strategy is to get an optimal caching strategy and a lower limit to which we can compare the LRU and LFU strategies.

## 8.5 Evaluation

We simulate 5 days of TV distribution in a simple scenario as described in section 8.3. There are 1000 viewers that, when active, request scheduled TV or time-shifted TV. The scheduled TV channels are distributed with multicast via a server; and all programs available for time-shifted viewing are also available from this node. The questions we address are: *how much bandwidth can we save by introducing a local cache (between the viewers and the server)? and how does the result depend on cache size and caching strategy in the local cache?*

Figure 8.7 shows the link load on the down link during the 5 simulated days. The top figure shows the case without a local cache, where all time-shifted TV are distributed in unicast streams from the core. The middle graph shows the link load when a 25 GB local cache is introduced. The bottom figure shows the result with a cache that is sufficiently large (250 GB) to hold all available time-shifted programs.

The graphs in figures 8.8 and 8.9 show the maximum and mean link loads for different cache sizes during the last three days of the simulations. Here we also see a comparison between different caching strategies. The steep slope of the curves show that introducing even a small cache can decrease the load on the down link considerably. The LFU strategy performs better than LRU and is also close to the lower limit set by the Clairvoyant caching strategy.

With a sufficiently large cache, with room for all available time-shifted TV programs, the traffic down from the core to the local cache is low, but not zero. Some viewers are watching the scheduled TV channels that are distributed with multicast from the core. There are also still some unicast transfers of time-shifted programs from the server. This is because, with the caching strategies investigated, a program is only distributed and cached if someone is requesting it. Scheduled programs that none of our 1000 viewers are watching (for instance during night when few viewers are active) are not distributed. If this program is later requested on-demand then, the first time, it is transfered with unicast from the server. This explains why, even with a sufficiently large cache, the link load down to the cache, can exceed that of 13 multicast channels (which would require 26 Mbps with the parameter setting used in the simulations).

The use and efficiency of the local cache strategy depend on the request pattern i.e what programs the viewers request and in what order. To compare the impact on the LFU and LRU strategies we did 10 different simulation runs for each cache size. The error bars in figure 8.10 and 8.11 show lower and

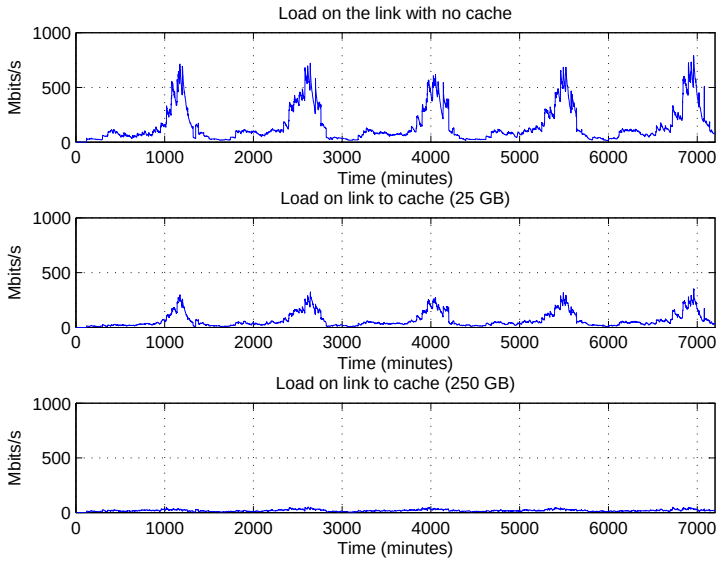



Figure 8.7: Link loads over five days for the cases: no cache, 25 GB cache, and 250 GB cache. Here the LFU caching strategy is used.

upper values for the resulting maximum and mean link loads on the down link.

The error bars are sometimes overlapping. A closer examination, as illustrated in figure 8.12, show that this is due to variation between simulation runs. With the same request pattern the LFU strategy always performs at least as good as the LRU strategy in our simulations.

## 8.6 Related Work

The recent growth and popularity of IPTV services have led to an increasing interest from researchers to measure and model IPTV viewing behavior. Cha et al. [14] present an extensive measurement study of viewing behavior including channel popularity and channel switching in an operational IPTV network. Ramos et al. [15] present work on constructing an IPTV workload model cap-

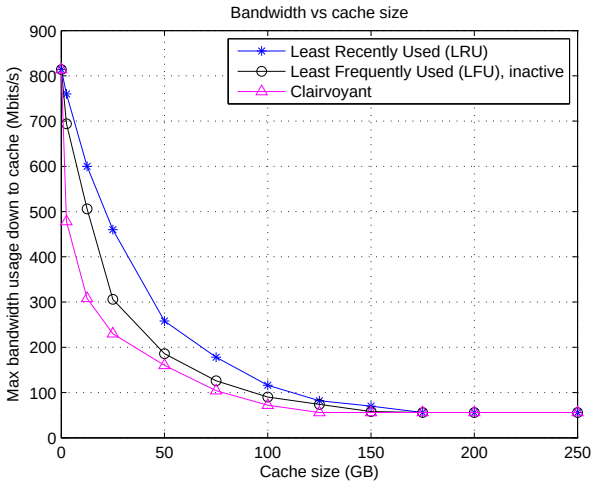



Figure 8.8: Comparing maximum bandwidth usage on the down link for different caching strategies and cache sizes.

turing the way viewers change channels and watch live TV. Qiu et al. model TV channel popularity [16] and user activities [17] in a large IPTV system and present the SimulWatch workload generator. Their model include set-top box on-times and off-times, channel popularity and channel switching. These studies are similar to ours in that they model IPTV viewer behavior – but they study traditional live TV, and model channel popularity and not the popularity of individual programs. We also simulate TV channels but our focus is on investigating time-shifted TV and caching, and for this the popularity of individual programs is a fundamental part of the model.

Yu et al. [18] measure and model user behavior and content access patterns in a large video-on-demand system. There has been a vast amount of research on different server scheduling techniques and proxy caching strategies and combinations of the two for video-on-demand systems and content distribution networks [1, 2, 3, 4, 5, 6]. These works are similar to ours in that they study methods for minimizing bandwidth requirements for media content distribution and investigate the trade-offs between network bandwidth and cache storage resources. Time-shifted TV has many similarities to VoD but for time-shifted TV the broadcasters' schedules decide when programs are released and

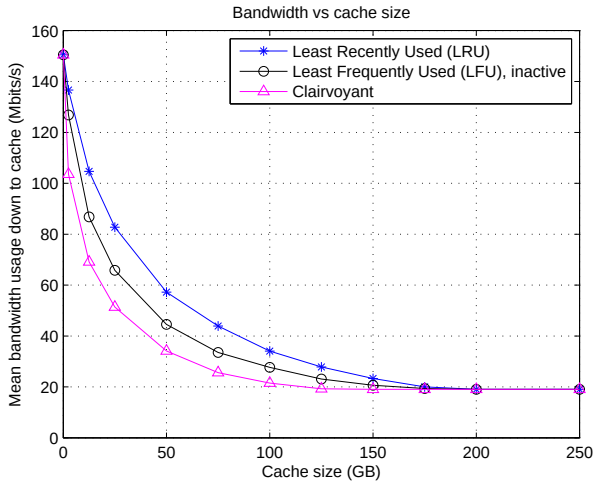



Figure 8.9: Comparing mean bandwidth usage on the down link for different caching strategies and cache sizes.

influence when and what people watch.

The work closest to ours are the studies by Wauters et al. [9, 10], Vleeschauwer et al. [8], and Krogfoss et al. [7]. They investigate the trade-off between bandwidth usage and storage in scenarios with time-shifted IPTV. But these studies have a more theoretical approach in that they do not use real TV schedules or TV statistics. Wauters et al. present an analytical model [9] and simulations [10] of time-shifted TV with a sliding-interval caching mechanism and co-operative caching. Vleeschauwer et al. [8] study a Catch-Up TV service where the viewers can select to watch the content at a time later than the original airing time. Based on observations from monitoring real TV program popularity they present a user behavior model and simulations where Poisson processes are used to generate the time when programs are aired (corresponding to the TV schedule) and the users' requests for a program. Their conclusions are consistent with ours that caching is needed to limit the otherwise enormous bandwidth requirements when new TV services are fully introduced. Krogfoss et al. [7] investigate several aspects of caching and optimization strategies for IPTV networks including network dimensioning and cache placement.

There are also related work that look at the larger picture of managing

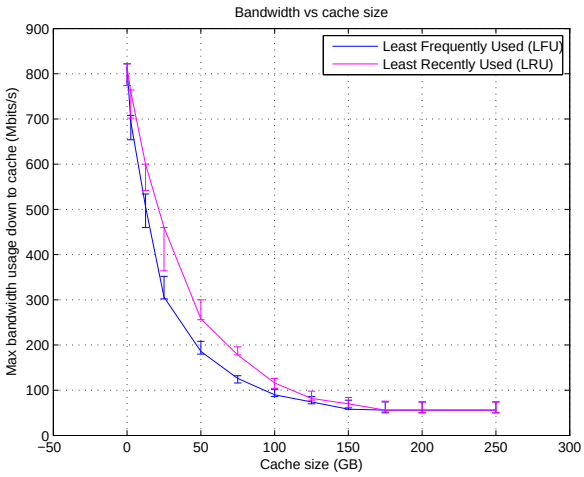



Figure 8.10: Maximum bandwidth usage for LFU and LRU with error bars (10 simulation runs).

a whole IPTV deployment. Mahimkar et al. [19] present work on performance diagnosis in a large IPTV network. Agrawal et al. [20] develop a general framework for planning an IPTV service deployment. Much research has also focused on peer-to-peer techniques for TV distribution [21, 22, 23] and VoD [24, 25, 26]. Cha et al. [21] analyze the TV viewing behavior in an IPTV system and explore how P2P-techniques can complement existing Telco-managed IPTV architectures.

## 8.7 Future work

IPTV with time-shift and the use of caching for IPTV are still at an initial stage of development. We have here studied the basic LRU and LFU caching algorithms. For future work there are more complex IPTV scenarios and IPTV caching strategies to investigate including co-operative caches, pre-caching during low traffic and more. Furthermore, the monetary cost of introducing memory into the network versus providing the bandwidth needed is important for operators.

There are also many possible refinements of the simulation model including

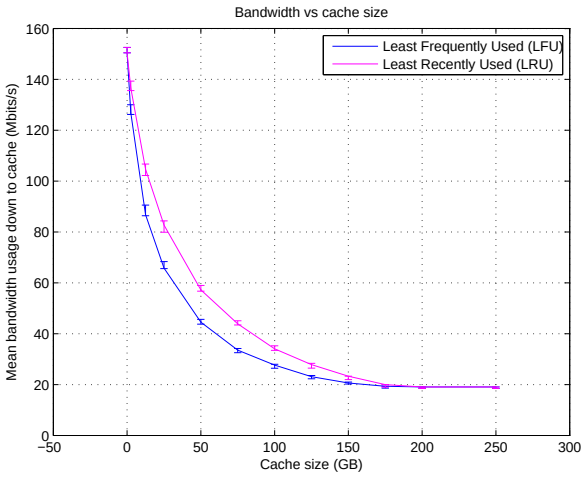



Figure 8.11: Mean bandwidth usage for LFU and LRU with error bars (10 simulation runs).

tuning parameters such as the popularity of time-shifted programs and introducing more complex viewer behavior.

As described in section 8.6 much research has been done on caching for video-on-demand. Time-shifted TV is something different in that we have an initial multicast distribution of all programs and that the broadcasters' schedules decide when programs are released and influence when and what people watch. Also, much of the work on caching for video-on-demand, including sliding-interval, prefix- and segment caching surveyed in [1], assume that only a small part of a program can be kept in memory. The current trend with memory prices going down makes it possible to put much larger caches into the network today than just a few years ago.

In this work we have assumed that all parts of a TV-program have the same popularity. When more detailed viewing statistics become available for time-shifted TV; and if it shows that the popularity of different parts of programs differ a lot, then it could be interesting to re-visit and evaluate more fine-grained segment-based caching algorithms also in the context of IPTV.

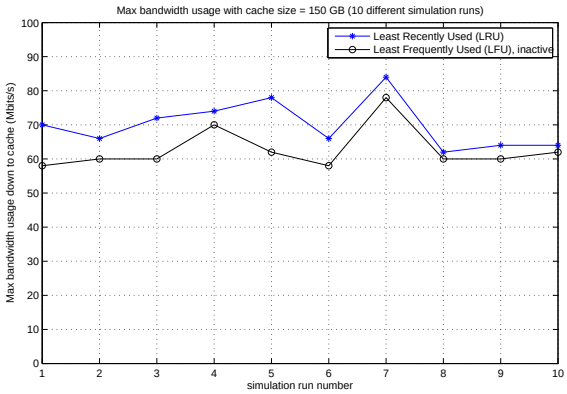



Figure 8.12: Detailed look on the maximum bandwidth usage for 10 simulation runs with LFU and LRU and cache size = 150 GB. The LFU strategy is always the better.

## 8.8 Conclusions

IPTV is now popular and widespread. Many telecom and broadband companies have become TV operators and distribute TV channels using IP multicast in their network. Operators also gradually introduce new services like time-shifted TV where the viewers can choose to watch the programs later after their first airing.

With a centralized system, unicast distribution of time-shifted programs, and hundred-thousands of subscribers, time-shifted IPTV distribution can be very bandwidth demanding. And since TV statistics show that most people are watching the same programs there can be a very large amount of copies of the same content distributed and putting load on the network. Caching the most popular programs closer to the viewers can significantly reduce the network load, as we show in this paper.

The effectiveness of caching depend on several factors including viewing behavior, request patterns and program popularity. For the development and evaluation of good caching strategies it is therefore important to develop realistic IPTV workload models that include the new time-shifted TV services.

In this paper we present a simple IPTV workload model, simulate IPTV distribution with time-shift, and show that caching can limit the bandwidth

requirements significantly.

## Acknowledgment

This work has been performed within the SICS Center for Networked Systems funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab Systems, TeliaSonera and T2Data.

# Bibliography


[1] Jiangchuan Liu and Jianliang Xu. Proxy Caching for Media Streaming Over the Internet. *IEEE Communications Magazine*, 42:88–94, 2004.

[2] D. Eager, M. Ferris, and M. Vernon. Optimized Regional Caching for On-Demand Data Delivery. In *Proceedings of Multimedia Computing and Networking (MMCN'99)*, San Jose, California, January 1999.

[3] S. Ramesh, I. Rhee, and K. Guo. Multicast with Cache (Mcache): An Adaptive Zero-Delay Video-on-Demand Service. In *Proceedings of IEEE INFOCOM'01*, Anchorage, Alaska, April 2001.

[4] C. Venkatramani, O. Verscheure, P. Frossard, and KW Lee. Optimal proxy management for multimedia streaming in content distribution networks. In *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video (NOSSDAV'02)*, pages 147–154, Miami, USA, 2002.

[5] O. Verscheure, C. Venkatramani, P. Frossard, and L. Amini. Joint server scheduling and proxy caching for video delivery. In *Proceedings of the Sixth International Workshop on Web Caching and Content Distribution*, Boston, USA, June 2001.

[6] B. Wang, S. Sen, M. Adler, and D. Towsley. Optimal proxy cache allocation for efficient streaming media distribution. In *Proceedings of INFOCOM'02*, New York, USA, June 2002.

[7] B. Krogfoss, L. Sofman, and A. Agrawal. Caching Architectures and Optimization Strategies for IPTV Networks. *Bell Labs Technical Journal*, 13:13–28, 2008.

[8] D. De Vleeschauwer, Z. Avramova, S. Wittevrongel, and H. Brueel. Transport Capacity for a Catch-up Television Service. In *Proceedings of EuroITV'09*, pages 161–170, Leuven, Belgium, June 2009.

[9] T. Wauters, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, and E. Six. Co-operative Proxy Caching Algorithms for Time-Shifted IPTV Services. In *Proceedings of 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pages 379–386, Dubrovnik, Croatia, September 2006.

[10] T. Wauters, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, and E. Six. Management of time-shifted IPTV services through transparent proxy deployment. In *Proceedings of IEEE Globecom 2006*, San Francisco, USA, November 2006.

[11] Mediamätning i Skandinavien (MMS). On–line: http://www.mms.se.

[12] Nielsen Audience Measurement. On–line: http://en-us.nielsen.com/.

[13] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, New York, 1991.

[14] Meeyoung Cha, Pablo Rodriguez, Jon Crowcroft, Sue Moon, and Xavier Amatriain. Watching Television Over an IP Network. In *Proceedings of Internet Measurement Conference(IMC)*, pages 71–84, Greece, October 2008.

[15] Fernando M.V. Ramos, Fei Song, Pablo Rodriguez, Richard Gibbens, Jon Crowcroft, and Ian H. White. Constructing an IPTV Workload Model. In *Proceedings of SIGCOMM, Poster session*, Barcelona, Spain, August 2009.

[16] Tongqing Qiu, Zihui Ge, Seungjoon Lee, Jia Wang, Qi Zhao, and Jun Xu. Modeling Channel Popularity Dynamics in a Large IPTV System. In *Proceedings of SIGMETRICS*, pages 275–286, Seattle, USA, June 2009.

[17] Tongqing Qiu, Zihui Ge, Seungjoon Lee, Jia Wang, Jun Xu, and Qi Zhao. Modeling User Activities in a Large IPTV System. In *Proceedings of Internet Measurement Conference*, pages 430–441, Chicago, USA, November 2009.

[18] H. Yu, D. Zheng, B. Zhao, and W. Zheng. Understanding User Behavior in Large-Scale Video-on-Demand Systems. In *Proceedings of EuroSys2006*, pages 333–344, Leuven, Belgium, 2006.

[19] Ajay Mahimkar, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Qi Zhao. Towards Automated Performance Diagnosis in a Large IPTV Network. In *Proceedings of SIGCOMM*, pages 231–242, Barcelona, Spain, August 2009.

[20] D. Agrawal, M. Beigi, C. Bisdikian, and K. Lee. Planning and Managing the IPTV Service Deployment. In *Proceedings of 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)*, pages 353–362, Munich Germany, May 2007.

[21] M. Cha, P. Rodriguez, S. Moon, and J. Crowcraft. On Next-Generation Telco-Managed P2P TV Architectures. In *Proceedings of International workshop on Peer-To-Peer Systems (IPTPS)*, 2008.

[22] X. Hei, C. Liang, J. Liang, Y. Liu, and K.W. Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9:1672–1687, December 2007.

[23] X. Zhang, J. Liu, B. Li, and T. Yum. Coolstreaming/donet: A data-driven overlay network for efficient live media streaming. In *Proceedings of IEEE INFOCOM'05*, Miami, FL, USA, March 2005.

[24] C. Huang, J. Li, and K.W. Ross. Can Internet VoD be Profitable? In *Proceedings of ACM Sigcomm 2007*, pages 133–144, Kyoto, Japan, 2007.

[25] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang. Challenges, Design and Analysis of a Large-scale P2P-VoD System. In *Proceedings of ACM SIGCOMM 2008*, pages 375–388, Seattle, USA, August 2008.

[26] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello. Push-to-Peer Video-on-Demand System: Design and Evaluation. *IEEE Journal on Selected Areas in Communications*, 25:1706–1716, December 2007.

# Chapter 9

# Paper D:
# Caching for IPTV distribution with time-shift

Henrik Abrahamsson and Mats Björkman.
In *International conference on Computing, Networking & Communications (ICNC'13)*, 28-31 January 2013, San Diego, USA.

**Abstract**

Today video and TV distribution dominate Internet traffic and the increasing demand for high-bandwidth multimedia services puts pressure on Internet service providers. In this paper we simulate TV distribution with time-shift and investigate the effect of introducing a local cache close to the viewers. We study what impact TV program popularity, program set size, cache replacement policy and other factors have on the caching efficiency. The simulation results show that introducing a local cache close to the viewers significantly reduces the network load from TV-on-Demand services. By caching 4% of the program volume we can decrease the peak load during prime time by almost 50%. We also show that the TV program type and how program popularity changes over time can have a big influence on cache hit ratios and the resulting link loads.

## 9.1 Introduction

Many telecom and broadband companies have become TV operators. They distribute TV channels using IP multicast in their networks but also gradually introduce new services like time-shifted TV (or TV-on-Demand) where viewers can choose to watch the programs later, after its scheduled time. Distributing individual TV streams to each viewer requires a lot of bandwidth and server capacity resulting in a big challenge for TV operators.

One way to reduce the network load is to cache popular content closer to the users. Caching is a well studied technique for web content and video [1, 2], but TV is different in many ways.

The potential for caching depends on several factors including user behaviour and content popularity. The new TV-on-Demand services have many similarities with Video-on-Demand systems but there are also some clear differences. In many traditional Video-on-Demand systems there are only a few new releases of movies every week. For TV distribution with time-shift the TV schedule with many ongoing channels gives a constant inflow of new programs that become available for on-demand requests. The program popularity is also different. Many TV programs have a very short lifespan. For instance news programs and weather forecasts quickly become outdated and lose their popularity as soon as a more recent report is made available.

It is important to understand what impact these characteristics of the new IPTV services have on caching. For the development and evaluation of good caching strategies it is also important to develop realistic IPTV workload models that include the new time-shifted TV services and how popularity changes over time.

In this paper we use an empirical IPTV workload model to simulate IPTV distribution with time-shift. The simulations are based on real TV schedules, and statistics about TV program popularity and viewer activity.

The contributions of this paper are: we show that a comparatively small local cache can be used to significantly reduce the peak link loads for TV distribution with time-shift. We also show that in addition to cache size and cache replacement policy, TV program type and how program popularity changes over time can have a big influence on cache hit ratio and the resulting link loads.

The rest of the paper is structured as follows: In Section 9.2 we describe our data set and give examples of TV viewing behaviour both from traditional linear TV and time-shifted TV. In Section 9.3 we describe the IPTV simulator and the simulation scenario. The simulation results, showing the impact of

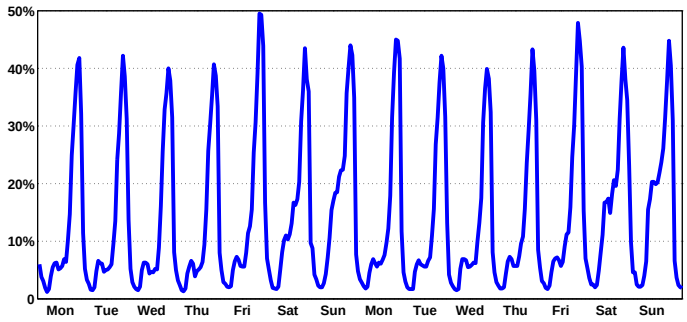



Figure 9.1: Fraction of active TV viewers over two weeks. Predictable daily and weekly variations in viewer behaviour.

TV program popularity, cache size, cache replacement policy, and program set size, are in Section 9.4. Related work is in Section 9.5, and we conclude the paper with a discussion in Section 9.6.

## 9.2 On TV viewing behaviour

### 9.2.1 Traditional linear TV

We use a data set from traditional TV with 13 channels over 28 days from Mediamätning i Skandinavien (MMS) [3]. MMS together with Nielsen Audience Measurement [4] measure the viewing habits of the TV audience in Sweden. The measurements are done using a so called People Meter system where the viewing habits of sample households are logged using electronic meters connected to the remote control.

Our data set includes 11635 TV programs from the most popular TV channels in Sweden. For each TV program we extracted the time it was scheduled, its length and the number of viewers. There is a large variation in the number of viewers between different programs. The median number of viewers of the programs in our data set is 27000. The maximum number of viewers of a program is almost 3.3 millions.

The data set also gives us information about the total number of viewers that are active and watch TV at a given time. Figure 9.1 shows the fraction of the viewers that are active and how it varies over the first two weeks of the data

set. We can see predictable daily and weekly variations in viewer behaviour. There are small increases in viewer activity each morning and there are distinct peaks during prime time every evening when up to 50% of the population is watching TV. As expected, Friday and Saturday evenings are the times when most people are watching TV, and we can also see that during the weekend more people are watching TV during daytime.

### 9.2.2 Time-shifted TV

For TV programs that are available on-demand, popularity declines with time. In a TV-on-Demand system, there is also a constant inflow of scheduled TV programs that become available on-demand. Therefore it is not the same programs that are the most popular day after day. Figure 9.2 shows two examples of how the number of viewers and the rank of programs decrease with time. The examples come from the Dutch TV-on-Demand site Uitzending Gemist [5]. Many programs such as news programs and weather forecasts quickly become outdated and lose their popularity when available on-demand. Other programs, typically drama TV-shows, retain interest from some viewers even a long time after their first release and initial peak in popularity. This categorization is for instance described by Avramova et al. [6] that study and model the popularity evolution of on-demand programs.

## 9.3 Simulation of IPTV with time-shift

In order to simulate IPTV distribution and evaluate caching strategies we use an empirical model based on the data set described in Section 9.2.1. We simulate IPTV distribution on the time scale of minutes. We have a TV schedule with 13 channels over 4 weeks and statistics about viewer activity and the popularity of the TV programs.

An earlier version of the simulator was presented in [7], and the reader is referred to that paper for details of the simulator that are not included here.

In the earlier version in [7] the TV programs were only available on-demand for a short time (24 hours) and they kept their original popularity. An important extension in the current version is that we simulate how program popularity changes over time and how this depends on program type. This is described in Section 9.3.3.

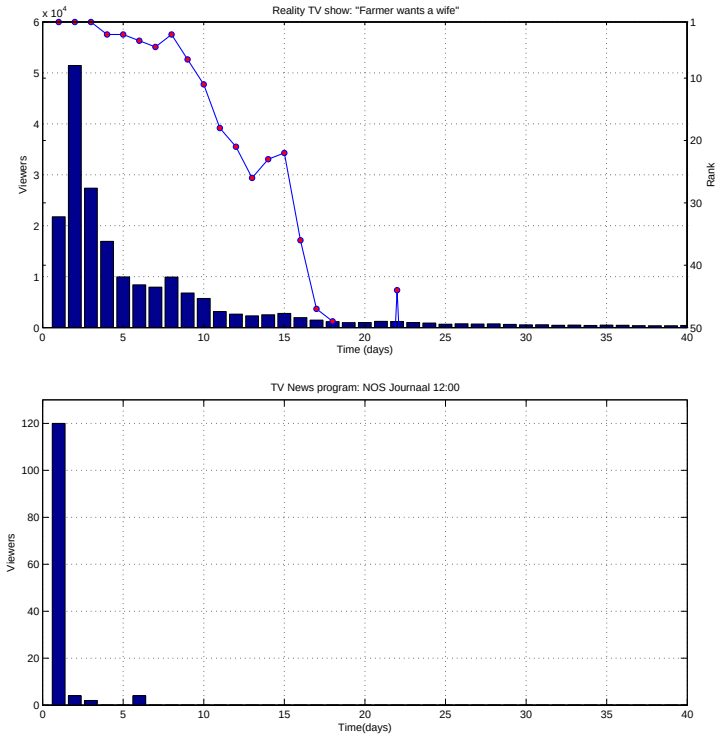



Figure 9.2: The top figure shows the number of viewers per day and the rank for an episode of the reality TV show *Farmer wants a wife*. The bar graph shows viewers per day with the scale on the Y-axis shown to the left. The plotted line shows the rank with the scale on the Y-axis shown to the right. The bottom figure shows the number of viewers per day for a TV news program. The figures show how the popularity changes over 40 days after the live broadcast. The popularity of a news program quickly declines.

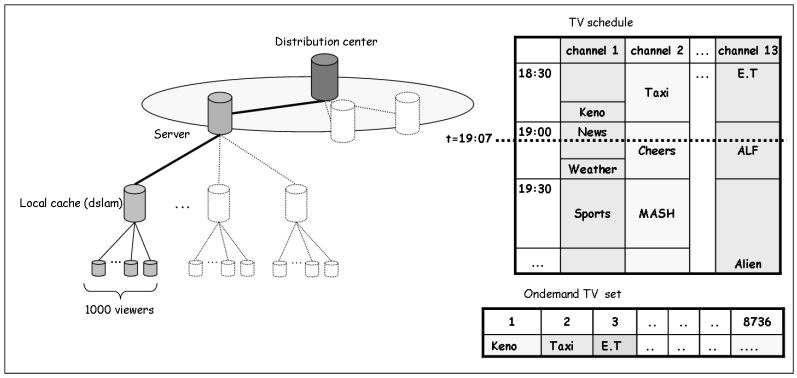



Figure 9.3: IPTV simulation scenario.

### 9.3.1 Network model and simulation scenario

We have a scenario with one branch of an IPTV network topology (as shown in Figure 9.3) with one server and one thousand viewers (TV set-top boxes). We simulate viewer requests for TV-programs and study the effect of introducing a local cache (in the DSLAM), the importance of cache size and cache replacement policy at this node, and the significance of TV program popularity and of the size of the set of available on-demand programs.

The scheduled TV channels are distributed with multicast and all programs then also become available for time-shifted viewing. We assume that all programs are distributed to the TV server and that all programs are stored there as long as they are available for time-shifted viewing. For the local cache it is different: what is stored in a local cache at a given moment in time depend on the size of the cache, the caching strategy in use, and what programs the viewers under the cache have chosen to watch (the request pattern). We monitor the cache hit ratio and the load on the link from the server to the local cache and we investigate how these change for different parameter settings.

### 9.3.2 TV viewers

In the simulator we follow the graph from the viewer statistics in Figure 9.1 closely and in each time step adjust the fraction of the viewers that are active and watch TV. A viewer that is activated chooses either to join the distribution of an ongoing scheduled program or to request one of the time-shifted

Table 9.1: Simulation parameters.

| | |
|---|---|
| Number of viewers | 1000 |
| Number of TV channels | 13 |
| Number of TV programs | 11635 |
| Programs available time-shifted | 21 days |
| Simulated time | 28 days |
| Scheduled TV/Time-shifted TV | 50/50 |
| TV stream bit rate | 2 Mbps |
| Total program volume (21 days) | 5064 GB |

programs that are available on-demand. The particular program to watch is chosen randomly following the empirical probability distribution for the popularity of the currently available programs. Table 9.1 shows the parameter settings for the simulations we present in this paper. If a viewer requests a scheduled program, and none of its neighbors is watching this channel, then a multicast stream is added to the load on the link down from the server to the local cache. If someone is already watching the channel, then the new viewer joins the ongoing multicast distribution and no additional load is added to the down link. Requests for time-shifted programs first go to the local cache. If the program is not available there, it is instead transferred with unicast from the server adding 2 Mbps to the load on the down link to the local cache.

### 9.3.3 TV programs

We step through, minute by minute, the TV schedule with 13 channels over 28 days. In each time step we update the set of programs. We also re-calculate the relative popularity of each program using the number of viewers of each scheduled program that we got from the input data set. The latter sets the probability that a simulated viewer will choose to watch a particular TV program.

The set of time-shifted programs is also updated in each step of the simulation. All scheduled programs goes into the set of time-shifted programs and can be requested on-demand. The time interval that the programs are available on-demand decides the size of the set of available programs. This a tunable parameter in the simulation that we investigate in Section 9.4.2. As a default value we let the programs be available on-demand for three weeks. With this parameter setting we have a steady state after 21 days when programs are deleted from the set of time-shifted programs in the same pace as new ones are sched-

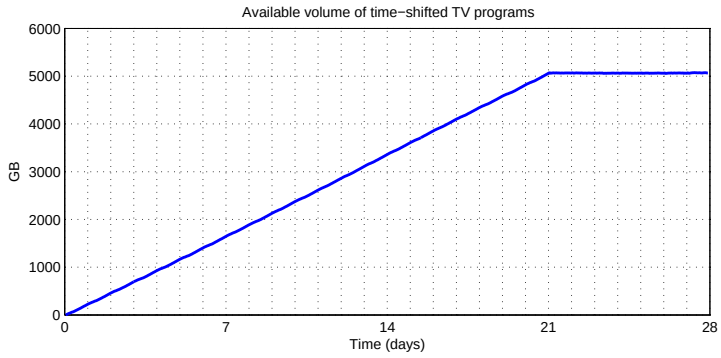



Figure 9.4: Available volume of time-shifted programs over 28 simulated days.

uled and introduced. Figure 9.4 shows the available volume of time-shifted programs over 28 simulated days. When evaluating the effect of caching in Section 9.4 we only consider the last simulated week. There are then on average 8736 time-shifted programs to choose between and a program volume of 5064 GB.

Each time-shifted program has a value of popularity which determines the probability that a viewer will choose to watch this particular program. This is initially set to the same value as the program had when scheduled (which is the number of viewers the program had in the input data set). But the popularity of a time-shifted program declines with time. Inspired by the work by Avramova et al [6] we investigated different functions for how TV program popularity changes. We categorized each program in our data set as either *News*-like or *Drama*. Programs such as news, business and weather reports, sports, game shows, and morning shows focusing on current events, were all classified into the News category of programs that quickly decrease in popularity. Other programs including movies, TV-series, and documentaries were classified into the Drama category with more slowly declining popularity. The mix of programs differs a lot between different channels. Some offers a lot of news programs, others have only movies and documentaries. With the 13 channels in our dataset 68% of the programs were sorted into the drama category of programs. In the simulations we let the news programs quickly lose their popularity when available on-demand, while the drama programs retain their interest over a longer time following the functions plotted in Figure 9.5.

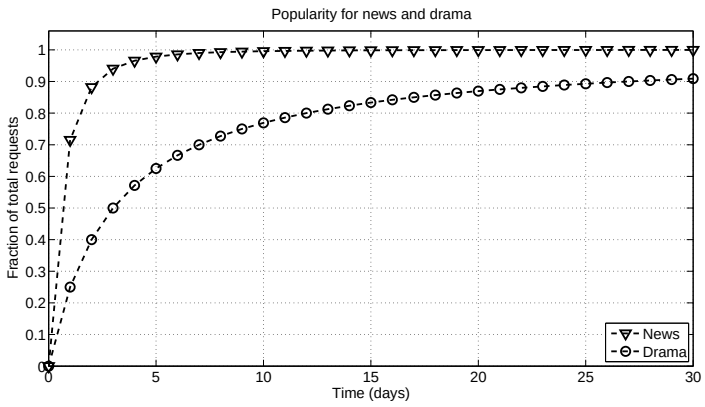



Figure 9.5: Popularity over time for programs categorized as news or drama.

### 9.3.4 Cache replacement policy

When the cache is full, and a new program part arrives, a cache replacement policy is needed to decide what should stay in the cache and what to delete. In this work we simulate and compare three classic policies: Least Recently Used (LRU), Least Frequently Used (LFU) and Clairvoyant.

With the LRU strategy we delete from the cache the program that has not been requested for the longest time. With LFU we discard the program that is requested least often. This could be done by counting the number of viewers that join the multicast distribution of a program and the number of on-demand requests. In the simulation we here use the known popularity of all programs, and delete the one with the least probability for being requested.

In the simulation we also implement a clairvoyant strategy with the ability to look into the future and delete the program part that will not be needed for the longest time. This is done by running the simulations twice. In the first run all viewer requests are logged; and in the second run this information is used to determine which program part that will not be asked for for the longest time. The purpose of this strategy is to get an optimal caching strategy and a upper limit to which we can compare the LRU and LFU strategies.

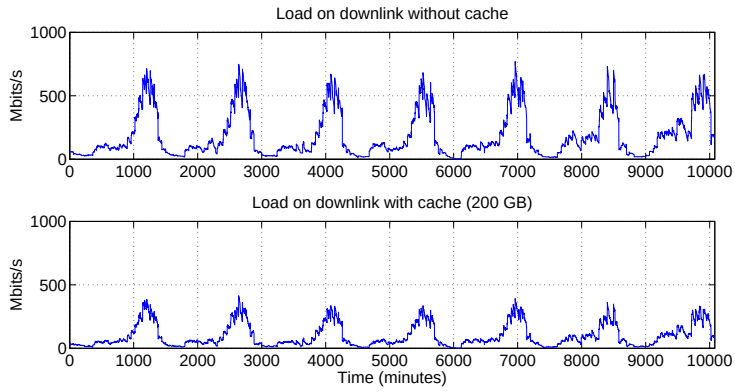



Figure 9.6: Load on downlink during 7 days for the cases with no cache (top) and with a 200GB cache (bottom).

## 9.4 Simulation results

We simulate TV distribution with time-shift in a simple scenario as described in Section 9.3. In Section 9.4.1 we study the effect of introducing a local cache, and how the resulting cache hit ratio and link load depend on cache size and cache replacement policy. Here we use the default parameter settings described in Section 9.3 where all programs from 13 channels are available on-demand for three weeks, and all programs are categorized as either news-like or drama and have a popularity that decreases over time correspondingly. In 9.4.2 we vary the time that the programs are available on-demand and study how this impacts on the caching efficiency. In 9.4.3 we investigate what significance program popularity have for caching.

### 9.4.1 Impact of cache size and cache replacement policy

Figure 9.6 shows the link load on the down link during the last 7 simulated days. The top figure shows the case without a local cache, where all time-shifted TV are distributed in unicast streams from the core. The bottom graph shows the link load when a 200 GB local cache is introduced using the LFU cache replacement policy. The volume of on-demand TV programs to choose from is on average 5064 GB during the last week of the simulation that we study here. A 200 GB cache can hold 4% of the available program volume

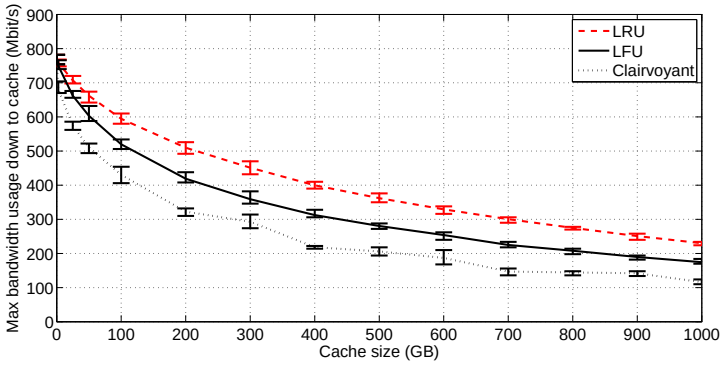



Figure 9.7: Comparing maximum bandwidth usage on the downlink for different cache sizes and cache replacement policies. The bars show minimum and maximum values from 5 simulation runs.

and on average it reduces the link load by 45.7%. The highest peak during primetime is reduced by 49.1% from 770 Mbps to 392 Mbps. The graphs in Figures 9.7 and 9.8 show the maximum link loads and cache hit ratios (CHR) for different cache sizes. Here we also see a comparison between different cache replacement policies.

We see in Figure 9.8 that LFU performs better than LRU. Caching 4% (200 GB) of the available programs with LFU gives a 50% hit rate. We can also see that there is still a lot of room for improvement up to the optimal clairvoyant replacement policy. With optimal caching (and a cache size of 200 GB) the hit ratio is increased from 50% to 62% compared to LFU.

### 9.4.2 Impact of on-demand time and program set size

If we change the time period that programs are available on-demand then we also change the size of the set of programs that are available for on-demand requests at a given time. Figure 9.9 shows the results for the cases when programs are available 1 day, 1 week, and 3 weeks. We can see that even though the program popularity declines with time, the time period that we let the programs be available on-demand has a big impact on the cache hit ratio. In these simulations the LFU cache replacement policy was used. For the case when the programs only are available 1 day there are on the average 417 on-demand

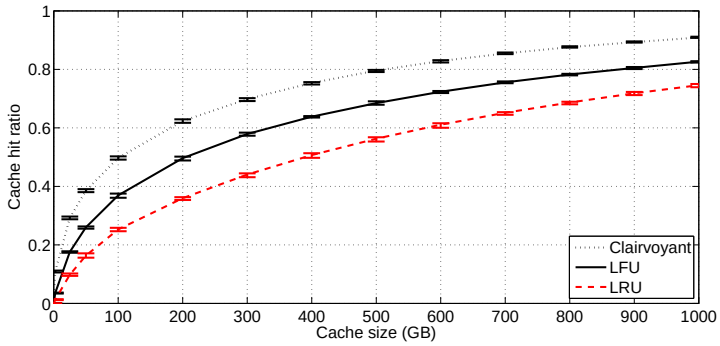



Figure 9.8: Cache hit ratio for different cache sizes and cache replacement policies. The total volume of available on-demand programs are on average 5064 GB. A cache size of 100 GB corresponds to 2%, and 500 GB is close to 10% of the available program volume.

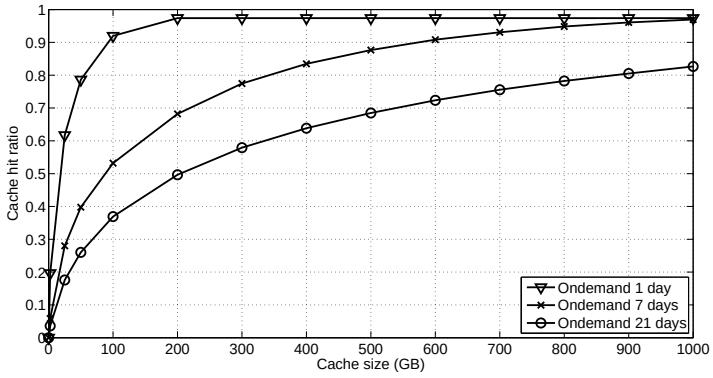



Figure 9.9: Comparing cache hit ratio for the cases when programs from 13 channels are available on-demand 1 day, 7 days and 21 days.

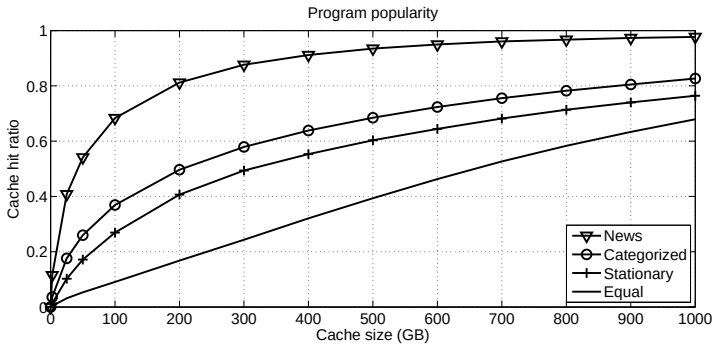



Figure 9.10: Influence of program popularity on cache hit ratio. The results for the set of categorized programs with decreasing popularity is compared to stationary program popularity, equal popularity, and with the case where all programs (as news programs) quickly decrease in popularity.

programs to choose between in our simulation scenario with an inflow from 13 channels.

### 9.4.3 Impact of program popularity

The popularity distribution of TV-programs is important for the usefulness of caching. This is also one aspect where TV with time-shift differs from traditional Video-on-Demand in that many popular TV programs (such as news and weather forecasts) have a very short life span.

In order to get an idea of what impact different aspects of program popularity have on the results we compare the cache hit ratio that we get with our work load model with the cases: stationary, equal, and news-like popularity. Figure 9.10 shows the results. In these simulations the LFU cache replacement policy was used. In the *equal* case the popularity of all programs are set to the same constant value. In the *stationary* case all programs keep their original values of popularity (which are the number of viewers the programs had when they were first aired) during the time they are available for on-demand requests. In our workload model we also categorize all programs and take into account that program popularity changes over time. A comparison between the stationary and categorized cases in the graph shows what impact this have on the results. In the *news* case we see what the cache hit ratio would be if all

programs were news-like and quickly decreased in popularity. For example, with a 200 GB cache (that can hold 4% of the program volume) we get a cache hit ratio of 17% for the *equal* case, we get a 50% hit ratio for our categorized workload model, and for the *news* case 81% of the requests can be handled by the local cache. This shows that the program type, or the type of TV channel that offers on-demand services, has a big influence on the caching results.

## 9.5 Related Work

The recent growth and popularity of IPTV services have led to an increasing interest from researchers to measure and model IPTV viewing behaviour. Cha et al. [8] study viewing behaviour including channel popularity and channel switching in an operational IPTV network. Qiu et al. model TV channel popularity [9] and user activities [10] in a large IPTV system and present the SimulWatch workload generator. These studies are similar to ours in that they model IPTV viewer behaviour – but they study traditional live TV, and model channel popularity and not the popularity of individual programs. We also simulate TV channels but our focus is on investigating time-shifted TV and caching, and for this the popularity of individual programs is a fundamental part of the model.

Gopalakrishnan et al. [11] measure and model in detail the interactive user behaviour in an IPTV environment, including how users fast-forward, pause and rewind to control their viewing. There are also many interesting studies of video popularity. Griwodz et al. [12] model long-term popularity of videos on the time scale of days based on VHS rental statistics. Tang et al. [13] analyse and model many aspects of media server access and implement a workload generator. Their model include both static and temporal file popularity and they distinguish between files with regular and news-like lifespan. Kang et al. [14] analyse workload on the Yahoo video sharing site. Gill et al. [15] and Cha et al. [16] present extensive studies of YouTube video sharing. Borghol et al. [17] study the popularity dynamics of Youtube videos. Yu et al. [18] study content access patterns in a large Video-on-Demand system. Lou et al. [19] study the popularity evolution of video files from a Chinese television station and use trace-driven simulation to evaluate caching in a p2p Video-on-Demand system. Dan and Carlsson [20] measure and analyse BitTorrent content popularity. Avramova et al. [6] study and model the popularity evolution of TV-on-Demand and video traces. Szabo and Huberman [21] predict the long-term popularity of online content at Digg and Youtube based on early measurements of user accesses.

Related work on caching include the work by Borst et al. [22] that study caching algorithms for content distribution networks. Wauters et al. [23], Vleeschauwer et al. [24], and Vleeschauwer and Laevens [25] use analytical models and simulations to study the performance of caching strategies in IPTV on-demand systems. These studies have a more theoretical approach and is in this sense complementary to our work. They do not use real TV schedules or TV statistics to run the simulations. Krogfoss et al. [26] also investigate several aspects of caching and optimization strategies for IPTV networks including network dimensioning and cache placement.

Much research has also focused on peer-to-peer techniques for TV and VoD including [27, 28, 29, 30].

## 9.6 Discussion

Our simulation results show that a comparatively small local cache can be used to significantly reduce the peak link loads during prime time. Caching 4% of the on-demand program volume gives a 50% hit rate with the LFU cache replacement policy. The simulation results also show that the program type, or the type of TV channel that offers on-demand services, has a big influence on the caching results. It matters whether we have news programs that quickly become outdated or movies that keep their popularity over a longer time.

For future work more complex IPTV scenarios and caching strategies may be considered. There are also several possible refinements of the simulation model such as separating the popularity of different segments of a time-shifted program and introducing more complex viewer behaviour. The large, predictable, daily variations in user demand described in Figure 9.1 means that it is important to have the right programs in the cache during prime time. It also makes pre-caching during low traffic an interesting area for further study. Furthermore, the monetary cost of introducing memory into the network versus providing the bandwidth needed is important for operators.

## Acknowledgment

This work has been performed within the SICS Center for Networked Systems funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab Systems, TeliaSonera, T2Data, Vendolocus and Peerialism.

# Bibliography


[1] Jia Wang. A Survey of Web Caching Schemes for the Internet. *ACM SIGCOMM Computer Communication Review*, 29:36–46, 1999.

[2] Jiangchuan Liu and Jianliang Xu. Proxy Caching for Media Streaming Over the Internet. *IEEE Communications Magazine*, 42:88–94, 2004.

[3] Mediamätning i Skandinavien (MMS). On–line: http://www.mms.se.

[4] Nielsen Audience Measurement. On–line: http://en-us.nielsen.com/.

[5] On–line: http://www.uitzendinggemist.nl/.

[6] Z. Avramova, S. Wittevrongel, H. Bruneel, and D. Vleeschauwer. Analysis and Modeling of Video Popularity Evolution in Various Online Video Content Systems: Power Law versus Exponential Decay. In *Proceedings of International Conference on Evolving Internet*, 2009.

[7] Henrik Abrahamsson and Mats Björkman. Simulation of IPTV caching strategies. In *Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS10)*, Ottawa, Canada, July 2010.

[8] Meeyoung Cha, Pablo Rodriguez, Jon Crowcroft, Sue Moon, and Xavier Amatriain. Watching Television Over an IP Network. In *Proceedings of Internet Measurement Conference(IMC)*, pages 71–84, Greece, October 2008.

[9] Tongqing Qiu, Zihui Ge, Seungjoon Lee, Jia Wang, Qi Zhao, and Jun Xu. Modeling Channel Popularity Dynamics in a Large IPTV System. In *Proceedings of SIGMETRICS*, pages 275–286, Seattle, USA, June 2009.

[10] Tongqing Qiu, Zihui Ge, Seungjoon Lee, Jia Wang, Jun Xu, and Qi Zhao. Modeling User Activities in a Large IPTV System. In *Proceedings of Internet Measurement Conference*, pages 430–441, Chicago, USA, November 2009.

[11] V. Gopalakrishnan, R. Jana, K. Ramakrishnan, D. Swayne, and V. Vaishampayan. Understanding Couch Potatoes: Measurement and Modeling of Interactive Usage of IPTV at large scale. In *Proceedings of Internet Measurement Conference (IMC'11)*, 2011.

[12] C. Griwodz, Michael Bär, and L. Wolf. Long-term Movie Popularity Models in Video-on-Demand Systems or The Life of an on-Demand Movie. In *Proceedings of the fifth ACM international conference on Multimedia*, pages 349 – 357, Seattle, USA, 1997.

[13] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat. Modeling and Generating Realistic Streaming Media Server Workloads. *Computer Networks*, 51:336–356, 2007.

[14] X. Kang, H. Zhang, G. Jiang, H. Chen, X. Meng, and K. Yoshihira. Measurement, Modeling, and Analysis of Internet Video Sharing Site Workload: A Case Study. In *Proceedings of IEEE Conference on Web Services*, 2008.

[15] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. YouTube Traffic Characterization: A view From the Edge. In *Proceedings of ACM SIGCOMM Internet Measurement Conference*, San Diego, USA, October 2007.

[16] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon. Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems. *IEEE Transactions on Networking*, 17:1357–1370, October 2009.

[17] Y. Borghol, S. Mitra, S. Ardon, N. Carlsson, D. Eager, and A. Mahanti. Characterizing and Modeling Popularity of User-generated Videos. In *Proceedings of IFIP International Symposium on Computer Performance, Modeling, Measurements and Evaluation (PERFORMANCE)*, 2011.

[18] H. Yu, D. Zheng, B. Zhao, and W. Zheng. Understanding User Behavior in Large-Scale Video-on-Demand Systems. In *Proceedings of EuroSys2006*, pages 333–344, Leuven, Belgium, 2006.

[19] J. Lou, Y. Tang, M. Zhang, and S. Yang. Characterizing User Behavior Model to Evaluate Hard Cache in Peer-to-Peer Based Video-on-demand Service. In *Proceedings of MMM'07*, pages 125–134, 2007.

[20] György Dan and Niklas Carlsson. Power-law Revisited: A Large Scale Measurement Study of P2P Content Popularity. In *Proceedings of the 9th international conference on Peer-to-peer systems (IPTPS'10)*, San Jose, USA, April 2010.

[21] G. Szabo and B. Huberman. Predicting the Popularity of Online Content. *Communications of the ACM*, 53, August 2010.

[22] Sem Borst, Varun Gupta, and Anwar Walid. Distributed caching algorithms for content distribution networks. In *Proceedings of INFOCOM'10*, San Diego, USA, 2010.

[23] T. Wauters, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, and E. Six. Co-operative Proxy Caching Algorithms for Time-Shifted IPTV Services. In *Proceedings of 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pages 379–386, Dubrovnik, Croatia, September 2006.

[24] D. De Vleeschauwer, Z. Avramova, S. Wittevrongel, and H. Brueel. Transport Capacity for a Catch-up Television Service. In *Proceedings of EuroITV'09*, pages 161–170, Leuven, Belgium, June 2009.

[25] Danny De Vleeschauwer and Koenraad Laevens. Performance of caching algorithms for IPTV on-demand services. *IEEE Transactions on broadcasting*, 55:491 – 501, 2009.

[26] B. Krogfoss, L. Sofman, and A. Agrawal. Caching Architectures and Optimization Strategies for IPTV Networks. *Bell Labs Technical Journal*, 13:13–28, 2008.

[27] C. Huang, J. Li, and K.W. Ross. Can Internet VoD be Profitable? In *Proceedings of ACM Sigcomm 2007*, pages 133–144, Kyoto, Japan, 2007.

[28] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang. Challenges, Design and Analysis of a Large-scale P2P-VoD System. In *Proceedings of ACM SIGCOMM 2008*, pages 375–388, Seattle, USA, August 2008.

[29] X. Hei, C. Liang, J. Liang, Y. Liu, and K.W. Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9:1672–1687, December 2007.

[30] Thomas Silverston, Olivier Fourmaux, Alessio Botta, Alberto Dainotti, Antonio Pescapé, Giorgio Ventre, and Kavé Salamatian. Traffic analysis of peer-to-peer IPTV communities. *Computer Networks*, 53(4):470–484, March 2009.

# Chapter 10

# Paper E: Program Popularity and Viewer Behaviour in a Large TV-on-Demand System

Henrik Abrahamsson and Mattias Nordmark.
In *Internet Measurement Conference (IMC'12)*, November 2012, Boston, USA.

**Abstract**

Today increasingly large volumes of TV and video are distributed over IP-networks and over the Internet. It is therefore essential for traffic and cache management to understand TV program popularity and access patterns in real networks.

In this paper we study access patterns in a large TV-on-Demand system over four months. We study user behaviour and program popularity and its impact on caching.

The demand varies a lot in daily and weekly cycles. There are large peaks in demand, especially on Friday and Saturday evenings, that need to be handled.

We see that the cacheability, the share of requests that are not first-time requests, is very high. Furthermore, there is a small set of programs that account for a large fraction of the requests. We also find that the share of requests for the top most popular programs grows during prime time, and the change rate among them decreases. This is important for caching. The cache hit ratio increases during prime time when the demand is the highest, and caching makes the biggest difference when it matters most.

We also study the popularity (in terms of number of requests and rank) of individual programs and how that changes over time. Also, we see that the type of programs offered determines what the access pattern will look like.

## 10.1 Introduction

Today increasingly large volumes of TV and video are distributed over IP-networks and over the Internet. Many telecom and broadband companies have become TV operators and distribute TV channels using IP multicast in their networks. The TV services also evolve, and are more and more changing towards TV-on-Demand and time-shifted viewing where the users can choose to watch the programs after its scheduled time. Distributing individual TV streams to each viewer requires a lot of bandwidth and server capacity. How to best use caching of popular content closer to the viewers is therefore an important issue to reduce network load.

In this paper we study access patterns in a large TV-on-Demand system over four months. We study user behaviour and program popularity and its impact on caching.

There are several studies of viewing behaviour in IPTV systems where traditional scheduled TV is distributed over IP networks [1], [2], [3]. This include studies of TV channel popularity and channel switching. Our work is different in that we look at TV-on-Demand where the viewers choose programs to watch outside of the TV schedule. The programs are not distributed using multicast but transferred with unicast streams to the viewers.

In this sense our work is closer to studies of content access patterns in traditional Video-on-Demand systems (VoD) [4]. But TV-on-Demand is different from traditional VoD in several ways. The TV-on-Demand service is more diverse. It is a mix of TV program libraries, time-shifted viewing, and rental video. Time-shifted viewing here means that the viewer can choose to watch ongoing scheduled TV-programs from the beginning. The TV schedule gives a large inflow of new programs each day. The programs available also come from a wide range of TV channels. There is a large variation in program types (news, drama, children's programs, movies, etc.) which each can have different access patterns. Many programs, like news and weather forecasts, also have a very short lifespan and are typically only interesting for a few hours.

The two main contributions of this paper are: (1) an investigation of program popularity and access patterns for TV and video on demand in a real network, (2) a trace-based study of caching. We characterize access patterns for different program categories, we show how program popularity changes over time and how this differs between different program types. We then use the request sequence in our data set for trace-driven simulation and study cache hit ratios for different cache sizes, cache replacement policies and population sizes.

Our main results are:

- The popularity (ranking) of rental movies, news, and TV shows changes over time in very different ways. News programs are often only requested for a few hours, movies are popular for months and increase in rank during weekends, TV shows increase in rank when the next episode is shown, children's programs are top ranked in the mornings and early evenings. This means that programs jumps in and out of the top 100 list. It also means that the *type* of content offered is essential for what the access pattern will look like.
- The program popularity conforms with the Pareto principle, or 80-20 rule. There is a small set of programs that account for a large fraction of the requests: the 2% most popular programs get 48% of the requests, and the 20% most popular programs get 84% of the requests.
- The share of requests for the top 100 most popular programs increases during prime time and the change among the top 100 decreases during prime time and during weekends when the demand is the highest.
- The cacheability is very high. The hit ratio with LRU is above 50% when caching 5% of the average daily demand, and the hit ratio increases during prime time when it is needed most.

The rest of the paper is structured as follows: In Section 10.2 we describe the TV-on-Demand service and introduce the data set. In Section 10.3 we study access patterns and the daily and hourly change in user interest. In Section 10.4 we look at the program popularity in more detail, how the access patterns differs between different program categories, and how the popularity of individual programs changes over time. In Section 10.5 we show what impact the access patterns has on cacheability and cache hit ratios. Related work is in Section 10.6, we discuss future work in Section 10.7 and conclude the paper in Section 10.8.

## 10.2 The data set

We study logs from the TeliaSonera TV-on-Demand service. The program selection is a mix from a wide range of TV channels (news, drama, children's programs, movies, etc.). It is a mix of TV program libraries, time-shifted viewing, and rental video.

Table 10.1: The data set in figures

| | Requests | Clients | Programs |
|---|---|---|---|
| Total (over 125 days) | 10294948 | 307347 | 89889 |
| Daily median | 80174 | 30232 | 7523 |
| Daily max | 121053 | 42451 | 8751 |
| Daily min | 56720 | 22194 | 6316 |

The TeliaSonera TV service also includes multicast distribution of traditional scheduled TV. Here we only study logs of on-demand requests but the TV schedule with many ongoing channels gives a constant inflow of new programs that become available for on-demand requests. In our data set, on average 8% of the programs each day have not been requested before.

The data set is a mysql database with logs from RTSP sessions where we for each session have: *<Timestamp, Length, ServerID, ClientID, AssetID>*.

The timestamp shows when the session ended and by subtracting the length of the session we get the time when the request arrived. The AssetID identifies what TV program is requested. For each asset, we also have additional out-of-band information about providers and program descriptions that help us categorize the programs into genres.

The data set is summarized in Table 10.1. It contains TV-on-Demand requests over 125 days between May 12th and September 13th 2011. During this period almost 90000 different programs were requested. The data set includes more than 300000 clients making more than ten million requests.

Figure 10.1 shows the number of requests, viewers, and programs per day. There are distinct weekly cycles where the number of active clients and the number of requests increase a lot during the weekends.

On average more than 30000 clients are active per day often increasing up to 40000 at the weekends. Some viewers are much more active than others and watch more TV programs. Viewers also subscribe to different TV packages and have access to different number of TV channels and program libraries. We can see that 5% of the viewers account for 41% of the requests and 20% of the viewers account for 75% of the requests. Figure 10.2 shows a log-log-plot of the number of requests per viewer. While many clients only watch a few on-demand programs per month, the most active viewer had more than 137 requests per day on average. Some of these sessions were 5-30 minutes long but many where short, jumping between different on-demand programs.

The clients in the data set are all in the same time zone and in the same

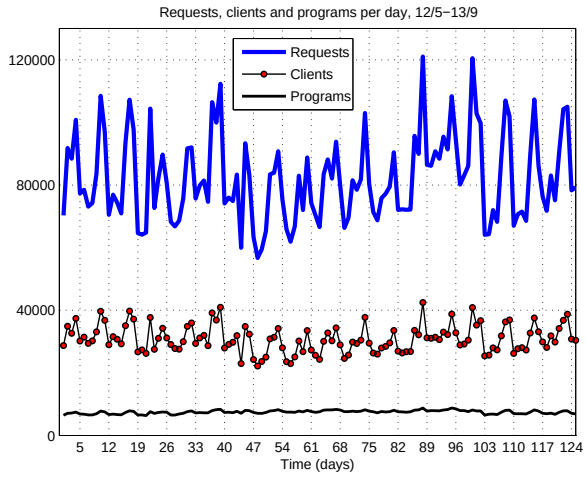



Figure 10.1: Number of requests, active clients, and distinct programs requested per day over 125 days. The grid shows weeks starting from Mondays.

geographical region. Later in Section 10.5, when looking at caching, we will also study smaller subsets of the population. We have one geographically close subset with 23304 clients in the same town. For the smaller populations in the study we randomly chose clients and include into sets of different size up to 10000. We will use the labels *region* (307347), *town* (23304), *rand10000*, *rand1000* etc. for the different populations.

On average 7523 different programs are requested per day. As expected, some programs are much more popular than others. On average the top 10 programs each day get 11% of the requests, the top 100 get 35%, and the top 1000 account for 71% of the requests. We will look at the program popularity in more detail in Sections 10.3 and 10.4.

## 10.3 Access patterns

### 10.3.1 Access pattern over a week

Figure 10.3 shows the number of requests per hour during one week from Thursday to Wednesday.

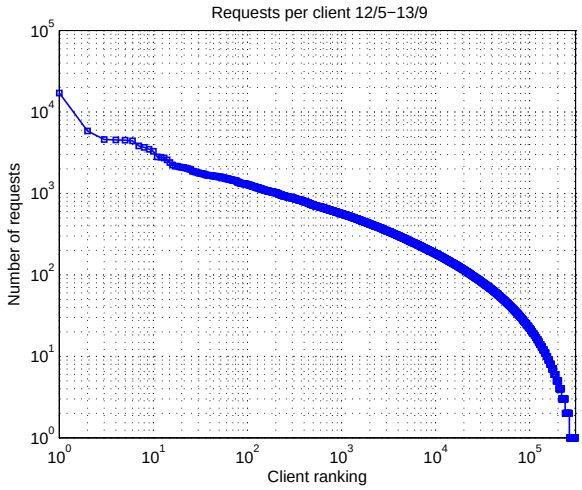



Figure 10.2: Log-log plot of requests per client over 125 days.

Table 10.2: The week 19-25/5 in figures

| | Requests | Clients | Programs |
|---|---|---|---|
| Total (over 7 days) | 585147 | 105698 | 14067 |
| Daily median | 76931 | 31542 | 6813 |
| Hourly median | 2626 | 1723 | 1190 |
| Hourly max | 16037 | 9019 | 2987 |
| Hourly min | 186 | 131 | 128 |

We can see here in detail the typical daily and weekly variation in demand. There are large, predictable peaks in demand in the evenings. The number of requests are often four times higher or more during the peak hour compared to the average demand during daytime. As expected, the number of requests are the highest on Friday and Saturday evening. The demand during daytime also increases during weekends.

The number of distinct programs requested per hour follows a similar pattern to that of the demand but the peaks are not as pronounced. The number of different programs requested often doubles in the evenings compared to daytime. In this particular week the median number of programs requested per

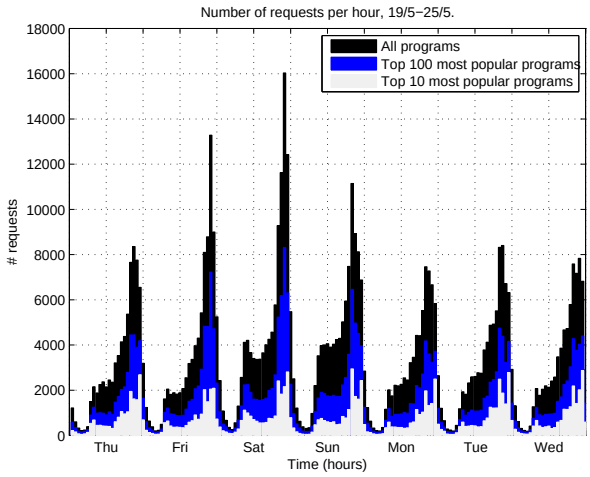



Figure 10.3: Number of requests per hour during a week. The grid points out the hours between 11:00-12:00 and 23:00-00:00 each day.

hour was 1190, the hourly maximum was 2987 programs (Saturday 21:00-22:00) and the minimum was 128 programs (Wednesday 04:00-05:00).

The bar plot in Figure 10.3 also shows the number of requests for the top 10 and top 100 most popular programs each hour. Figure 10.4 shows the share of requests per hour that the top 100 most popular programs account for. On an hourly basis the top 100 on average get 50% of the requests. The top 100 obviously have a large part of the traffic during night when not much more than 100 programs are requested. But more interestingly the top 100's share of requests also increase significantly during prime time. The number of different programs requested increases during the evenings and so a hundred programs constitute a smaller share of the requested program volume. But even so the top 100's share of requests increase significantly.

### 10.3.2 Daily and hourly change in user interest

Which programs are most popular change over time. On average 6 of the top 10 programs are replaced each day. Figure 10.5a shows the daily change among the top 100 and top 1000 most requested programs. Here we also see the

daily change among all requested programs. On average 73% of the requested programs are the same as yesterday. On average 56% of the top 100 and 42% of the top 1000 is different from the day before.

We notice in Figure 10.5a a weekly pattern with less change in top 100 during weekends (from Fridays to Saturdays and Saturdays to Sundays).

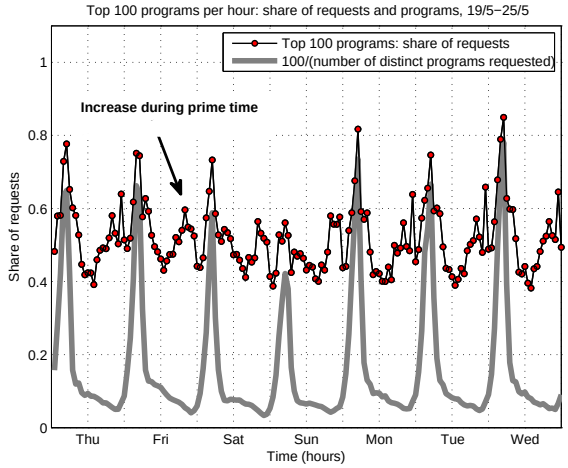
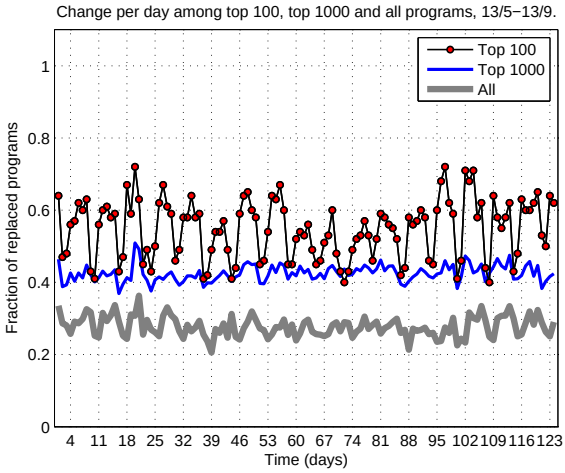


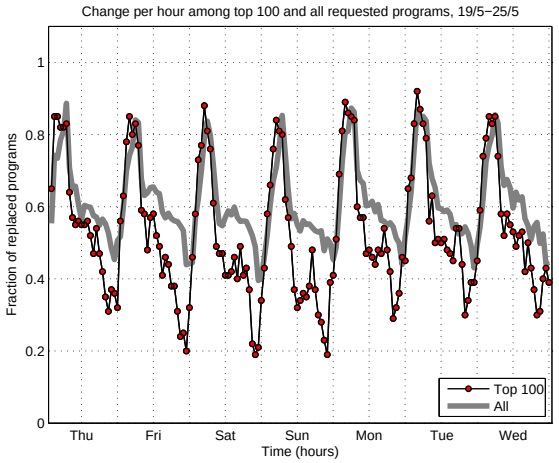

Figure 10.4: The figure shows the share of requests per hour that the top 100 most popular programs account for. It also shows the share of programs requested that a hundred programs comprises. The top 100's share of requests increases during night when few programs are requested but more interestingly it also increases during prime time when the demand is the highest. The grid points out the hours between 11:00-12:00 and 23:00-00:00 each day.

This suggests that what the most popular items will be is more predictable during weekends when the demand also is the highest. This is even clearer on an hourly basis.

Figure 10.5b shows the hourly change among the top 100 most requested programs. On average 51 out of the 100 most requested programs are the same as the hour before. But the amount of change in the top 100 varies from hour to hour in a distinct daily cycle. During night up to 92% of the top 100 programs are changed from one hour to the next. While during prime time (19:00 to 23:00) the top 100-list becomes much more stable with down to 19% change among the top 100 programs.




(a)




(b)

Figure 10.5: (a) Daily change among the top 100 and top 1000 most requested programs. (b) Hourly change among the top 100 most requested programs. The figures also shows the fraction of all programs that was not requested the day and hour before.

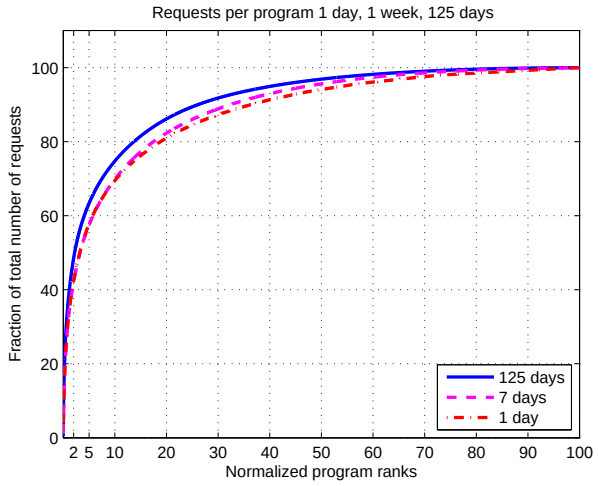



Figure 10.6: Cumulative distribution of requests to programs (1 day, 7 days and 125 days).

## 10.4 Program popularity

There is a small set of popular programs that account for a very large part of the requests. The Pareto principle, or the 80-20 rule, is often referred to when describing video popularity and the concentration of user interest towards a few popular programs [5], [4]. The users spread of requests across programs in the TV-on-Demand system conforms with this principle. The 20 % most popular programs account for more than 80% of the requests.

We calculated the number of requests for each program, sorted them in order of popularity, and plotted the cumulative distribution function shown in Figure 10.6. Here we can see the number of requests per program as a CDF-plot for 1 day, 1 week and for the entire 125-day period.

If we consider the entire 125-day period, then the 2% most popular programs account for 48% of the requests, the 10% most popular programs account for 74% of the requests, and the 20% most popular programs receive 84% of the requests. The figures are similar on daily and weekly basis as well.

This skewness in popularity for TV-on-Demand is somewhere in between what has been described in the literature for user-generated content and tradi-

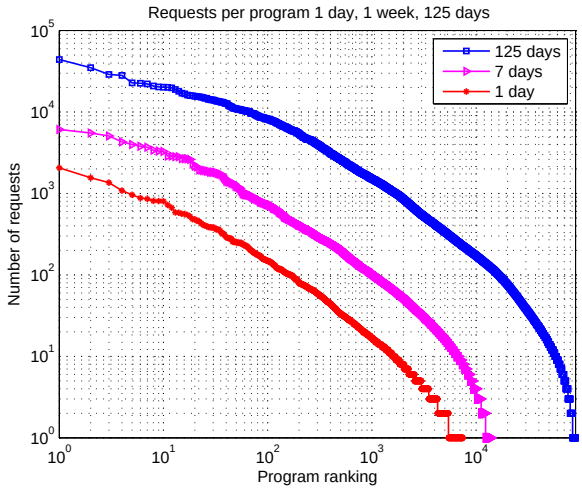



Figure 10.7: Log-log plot of requests per program (1 day, 7 days, 125 days).

tional Video-on-Demand systems. For Youtube traffic, investigated by Cha et al. [5], 10% of the videos accounted for 80% of the requests. In the chinese PowerInfo Video-on-Demand system described by Yu et al. [4], 10% of the videos accounted for 60% of the accesses. TV-on-Demand systems are more dynamic than traditional VoD systems with a large daily inflow of new content. As we will see in Section 10.4.2, there are programs that are popular for several weeks and accumulate a lot of requests, but there are also many programs that have a very short life-span and are only requested for a few hours.

Figure 10.7 shows the number of requests per program as a log-log plot. It shows the number of times that a program has been accessed versus the ranking of the program in the data set.

There are a large number of research papers that deal with the popularity distribution of web pages and video. Much of the debate concerns whether the distribution of requests is Zipf-like or not [6, 5, 7, 8, 4]. Here we do not try to fit the curve to a specific probability distribution. However, we note that the curve does not follow a straight line on the log-log scale. This implies that the distribution of TV-on-Demand requests does not follow a Zipf-like distribution.

The 80-20 rule, and the concentration of requests to a small set of programs is important for caching. This is independent of what exact probability

distribution best describes the access frequency. In Section 10.5 we perform trace-driven simulation, and directly use the sequence of requests to investigate the impact on caching.

### 10.4.1 Access patterns per program category

Different categories of programs have different access patterns. Figure 10.8 shows the number of requests per hour over two weeks for programs in four different categories: rental video, TV news, drama and children's programs. The figure demonstrates some clear and expected differences in access patterns.

The top figure shows the access pattern for rental movies. These are movies that a viewer can pay to access for 24 hours. We can see that movie rentals are concentrated over weekends with large peaks in demand during Friday and Saturday evenings.

For TV news the traditional TV schedule determines to a large extent also when the program is requested on demand. The TV news is scheduled daily at 19:00 and 22:00. At the same time it becomes available for time-shifted viewing and we can see that most requests are close to these times.

For the other two categories we note that the TV reality and drama shows are watched during the daytime to a larger degree than other programs. We also see that the children's programs have peaks in demand in the mornings and early evenings. This is especially true for weekends.

### 10.4.2 Access patterns for individual programs: how program popularity changes over time

The popularity of a program changes with time and the demand pattern varies depending on the program type. Figure 10.9 shows the number of requests per day for 20 different programs over 125 days.

The top figures show the most requested rental movies and TV news programs in the data set. For each movie we can see a slow decline in popularity over time. The movies are requested many weeks after their premiere. There is also a clear weekly pattern with peaks in demand at the weekends. For TV news programs the access patterns are very different compared to movies. A news program is mostly requested the first evening and then quickly becomes outdated and loses its popularity when available on demand.

The figures at the bottom show the access pattern for five episodes of a daily TV reality show and five episodes of a weekly home improvement show. We can see that the request patterns for different episodes of the same show

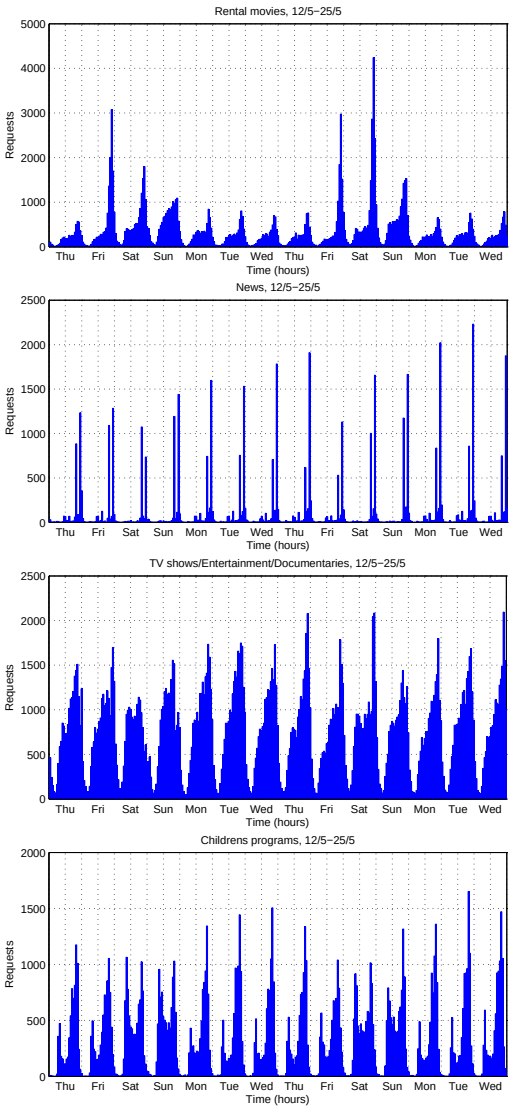



Figure 10.8: Access patterns per program category: rental movies, TV news, TV drama, and children's programs. Requests per hour over two weeks. The grid points out the hours between 11:00-12:00 and 23:00-00:00 each day.

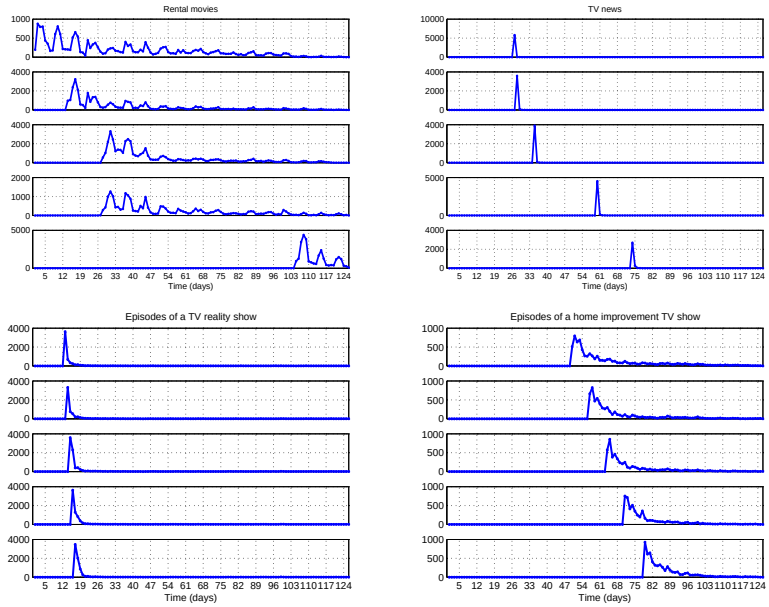



Figure 10.9: Requests per day for 5 movies, TV news programs and episodes of a reality show and a home improvement show.

are surprisingly similar. For the reality show we can also see that after the initial peak in demand the program popularity quickly decline with time. The programs are requested daily also the following three months but there are often only a few requests per day.

### The life of a rental movie

Figure 10.10a shows the number of requests per day and the rank for a comedy-drama rental movie. We can see the decline in popularity over 16 weeks from the premiere. The figure also demonstrates the typical peaks in popularity for a rental movie during weekends where the number of requests increases and the program increase in rank.

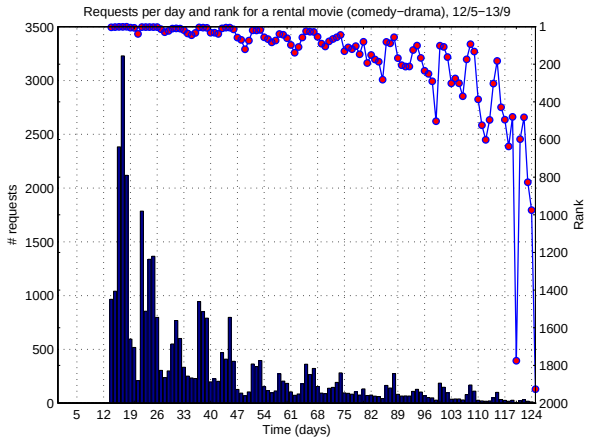
Figure 10.10b shows the change in rank in more detail among the top 100 most popular programs each day. We can see that the movie jumps in and out of the top 10 and top 100 lists a number of times. This has implications for the choice of caching strategy. It is essential to have the right programs in the cache at Friday and Saturday evenings when the total demand is the highest. If the replacement policy acts on popularity over a short time window the program might be evicted when popularity temporarily goes down during weekdays and the program will not immediately be available in the cache when the demand increases again next weekend.

Figure 10.10b also compare the rank for our movie among all programs with the rank among only rental movies. The movie is the number one most popular movie for 14 days in a row and it stays in the top 10 for one month and in the top 20 for two months.

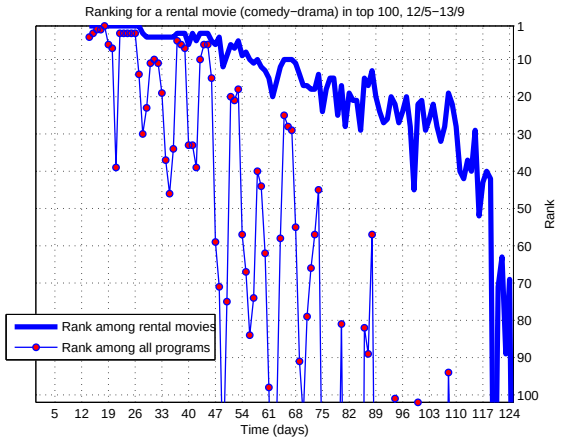
Figure 10.11 shows the rank and the number of requests per hour during the first week that the movie is available. The movie quickly climbs in rank and becomes one of the most popular programs. It is in the top ten during the evenings but the rank of the program sometimes drops during the daytime and during night. There are large, predictable daily variations in demand with peaks in the evenings. The number of requests increases significantly during Friday and Saturday.

### The life of a TV news program

TV news programs have a very short lifespan compared to movies. Figure 10.12 shows the rank and number of requests per hour for a news program that was sent live at 19:00.




(a)




(b)

Figure 10.10: (a) Requests per day and ranking for a rental movie (comedy-drama). The bar graph shows requests per day with the scale on the Y-axis shown to the left. The plotted line shows the ranking of the program among all other programs requested that day. The scale of the ranking is shown on the Y-axis to the right. The grid points out weeks starting on Mondays. (b) Detailed look at the rank among all programs (top 100) and among rental movies.

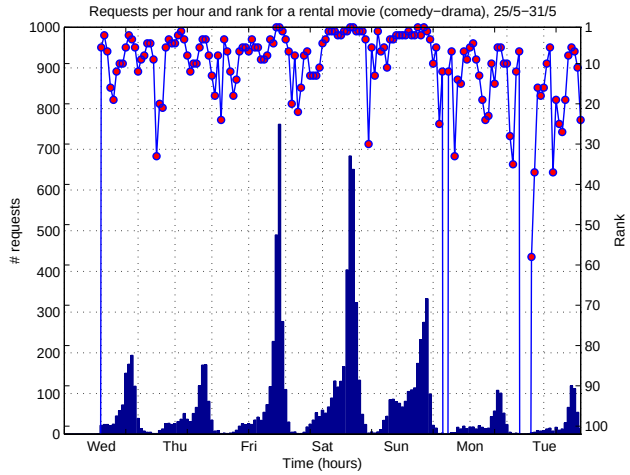



Figure 10.11: Requests per hour and ranking for a rental movie (comedy-drama). The grid points out the hours between 11:00-12:00 and 23:00-00:00.

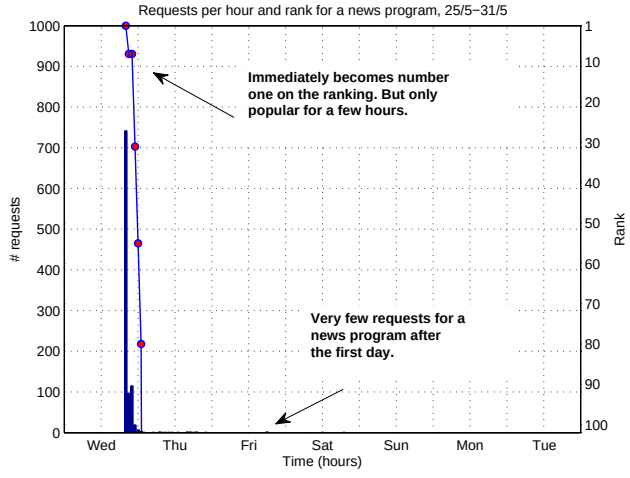



Figure 10.12: Requests per hour and ranking for a TV news program.

Most of the requests are the first hour when the program becomes available. The news program immediately becomes the most requested program that hour and number one on the ranking. The popularity then quickly declines. There are almost no requests at all for a news program after the first day. The access pattern is very different compared to what we see in Figure 10.11 for the simultaneously available movie.

### TV series and children's programs – periodic increase in popularity

The interest in a TV program usually decreases with time. But more often than not the popularity of a program can also increase temporarily or periodically. We saw in the previous sections that the number of requests for a program varies during the day and the week. We also saw for rental movies that the ranking increased during weekends.

Many TV shows are part of a series of programs. When the next episode is sent there is often also renewed interest for old episodes available on-demand. Figure 10.13 shows an example with the rank and number of started sessions per day for an episode of a weekly home improvement TV show. We can see that the program increase in rank every Thursday when the series is shown on the traditional scheduled TV.

Figure 10.14 shows the number of requests per day and the ranking for a cartoon. After the initial peak in interest the popularity decreases and remains at a steady level over the month when the program is available. This is different if we look at the ranking on an hourly basis. Figure 10.15 shows the ranking of the program per hour during the first week. The pattern is the same for the next three weeks as well. The program varies in popularity. It goes in and out of the top 100 list, often twice a day.

The number of requests for children's programs increases in the mornings and in the early evenings. This is a daily recurring pattern. Also, at these times of the day there is little demand for other TV-programs so few requests are needed to get into the top ranking.

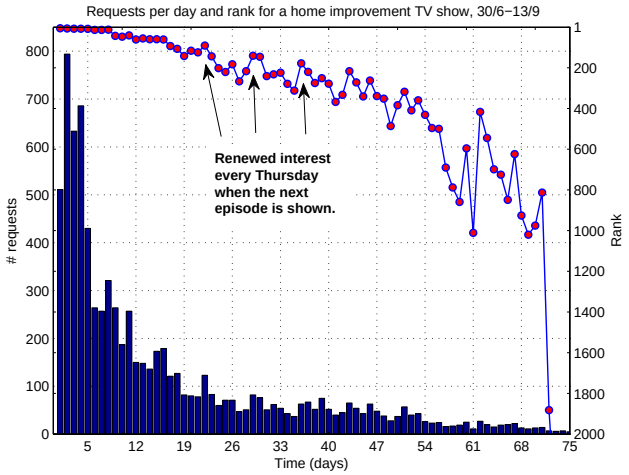



Figure 10.13: Requests per day and ranking for an episode of a weekly home improvement TV show. The grid points out weeks starting on Mondays.

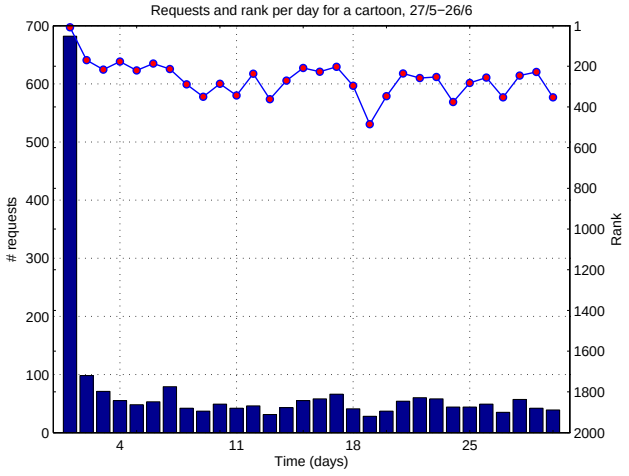



Figure 10.14: Requests per day and ranking for a cartoon.

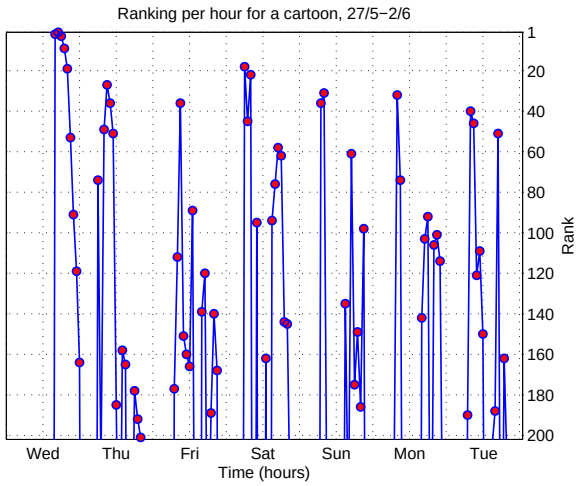



Figure 10.15: Ranking per hour for a cartoon. The grid points out the hours between 11:00-12:00 and 23:00-00:00 each day. The program jumps up and down in popularity. It always climbs to top 40 in the mornings and are often in top 100 in the early evenings.

## 10.5 Impact on caching

In previous sections we have seen many aspects of the access patterns in a TV-on-Demand system. In this section we study the impact on caching. We examine the proportion of requests that are not first-time requests for a program and therefore potentially could be served from a cache. We look at this for different population sizes and time periods.

We then use trace-driven simulation to investigate the cache friendliness of the workload with a limited cache size and the classic LRU and LFU cache replacement policies. We run the sequence of requests in our data set through caches of different size and look at the resulting cache hit ratios.

### 10.5.1 Cacheability

For on-demand caching, the first request for a program needs to go to the central server. But if we imagine an unlimited cache size then all other requests could potentially be served from the local cache. It is therefore interesting to examine the proportion of requests that are not first-time requests. We here call that *cacheability*.

We follow the definition of *cacheability* used by Ager et al. [9]. But our data set do not include information about program size so here we only consider requests. Cacheability is then the share of requests that are not first-time requests. If $k_i$ is the total number of requests for a program $i$ then the cacheability is $\sum_{i=1}^{n}(k_i - 1)/\sum_{i=1}^{n}(k_i)$, where $n$ is the number of programs.

The share of first time requests is very low in the TV-on-Demand system if we consider all clients over a long period of time. The cacheability over 125 days is: 99.13%.

In Figure 10.16 we also look at the cacheability per day and per hour and for populations of different size. The calculation of cacheability starts from the beginning of each time interval. It is not considering what have been requested the hour or day before. For all clients in the region during the week in Figure 10.16, the median cacheability per hour is 59%. However, there are large daily variations. During night many programs are requested only once and the cacheability is low. During Friday and Saturday evenings the cacheability is above 80%.

Figure 10.17 shows examples of cacheability over 125 days for smaller populations. For very small populations the probability that a viewer will choose a program that nobody else in the group has requested before is high. So the share of first-time requests is high and the cacheability is low. But we

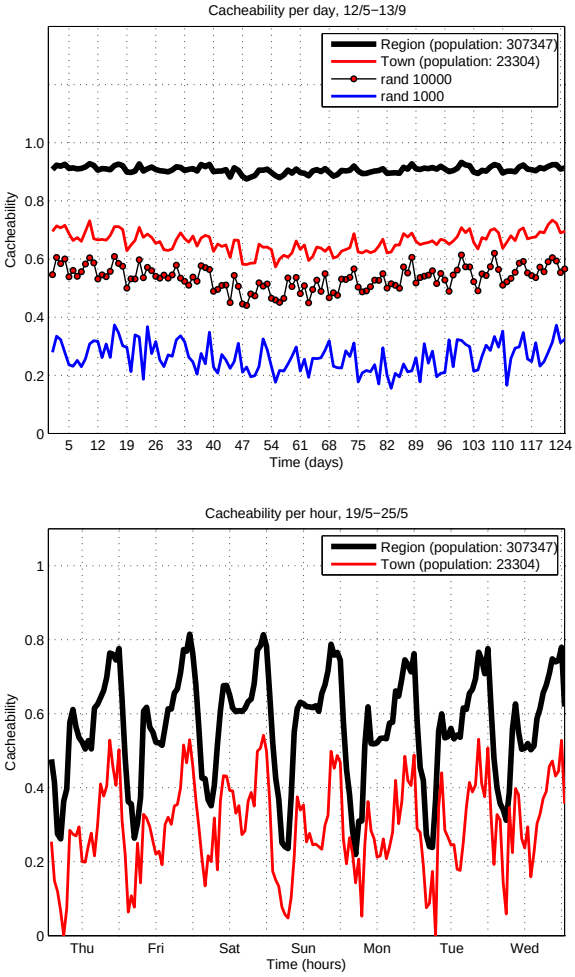



Figure 10.16: Cacheability per day and per hour. Comparison between different population sizes.

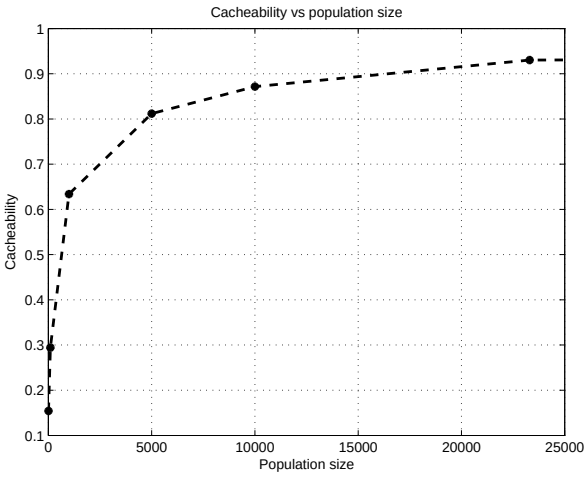



Figure 10.17: Example of cacheability versus population size.

see that already for groups of 1000 viewers the cacheability is above 60%. We calculated the cacheability for five different groups of 1000 viewers. The median was 63.9% and the group with lowest result had a cacheability of 61.7%.

### 10.5.2 Limited cache size

We saw in the previous section that the cacheabilty in the TV-on-Demand system is very high. But in practice there is a limited cache size. In order to investigate the cache friendliness of the TV-on-Demand workload we use trace-driven simulation. We run the sequence of requests in the data set through caches of different size and study the cache hit ratios for three classic caching policies:

**Least Recently Used (LRU):** with the LRU strategy we delete from the cache the program that has not been requested for the longest time.

**Least Frequently Used (LFU):** with LFU we discard the program that is requested least often. This is done by keeping track of the hit ratio for all programs currently in the cache (in cache LFU).

**Clairvoyant:** we also implement a clairvoyant strategy with the ability to look into the future and delete the program that will not be needed for the longest time. This is used for comparison to obtain an upper limit on the cache

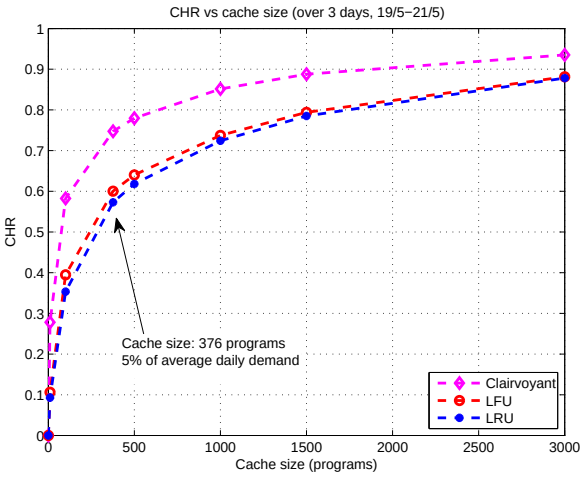



Figure 10.18: Cache hit ratio versus cache size, requests from all clients over 3 days. Comparison of the LRU, LFU and Clairvoyant replacement policies.

hit ratio. It is implemented by going through the traces twice. First, for each request of a program we look up and determine when the program will be requested next. This is then used in the simulation to determine what program should stay in the cache.

Figure 10.18 shows cache hit ratios for the LRU, LFU and Clairvoyant replacement policies for increasing cache sizes. The hit ratios are calculated over 3 days. The size of the programs are not taken into account. We calculate request (or program) hit rate and not the byte hit rate. The x-axis shows cache size in number of programs. The median number of distinct programs requested per day is 7523. To put the hit ratio and cache size in relation to the daily demand we therefore look specifically at cache sizes of 376 programs, which is 5% of average daily demand. We can see that caching 5% of the daily demand gives a hit ratio of 57% for LRU, 60% for LFU and 75% for the Clairvoyant replacement policy.

In Figure 10.18 we include the requests from all viewers. In Figure 10.19 we investigate the impact of population size. Here we use the LRU replacement policy and compare the cache hit ratios for populations of different size. For a cache size of 376 programs (5% of the daily demand) the town subset of

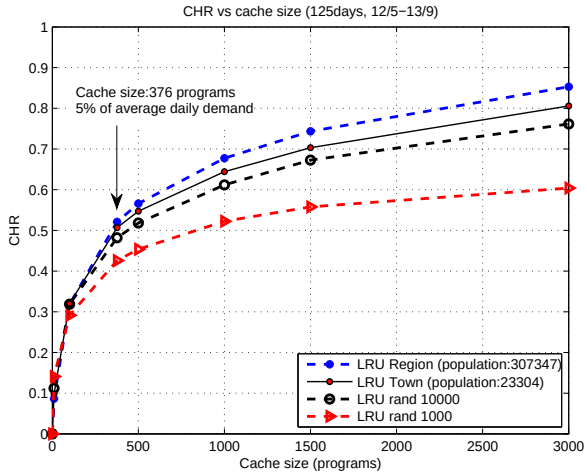



Figure 10.19: Cache hit ratio versus cache size. Comparison of cache hit ratios for populations of different size.

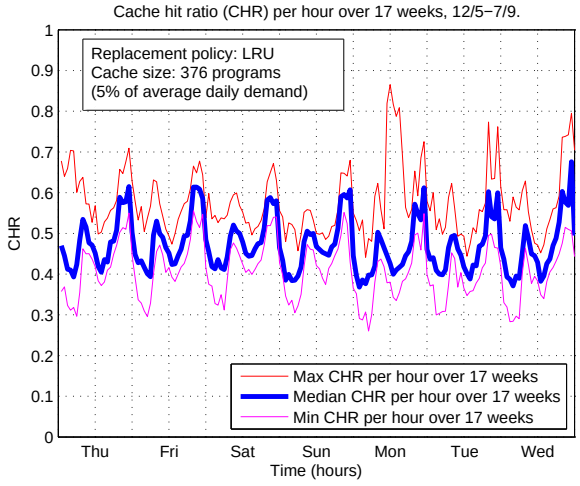



Figure 10.20: Cache hit ratio per hour over 17 weeks

23304 clients get a hit ratio of 51%. This is close to the result for the full set of clients. For the small population with 1000 clients we get a hit ratio of 43%. The cacheability for this particular population of 1000 viewers was $0.64$ over 125 days and we see the curve approaching that value at a cache size of 3000 programs.

The cache hit ratio also varies over time. Figure 10.20 shows hit ratio per hour for all viewers, the LRU replacement policy and a cache size of 376 programs. The hit ratio was calculated over 17 weeks and the figure shows the median (and max and min) value for each hour of the week. We can see that the cache hit ratio varies over the day and it increases when it is needed as most. During prime time, when there are the most requests, the hit ratio is over 60%.

From the results presented above we highlight three observations:

- The cacheability and the potential for caching is very high.
- The hit ratio with a simple LRU replacement policy is above 50% when caching 5% of the average daily demand.
- The hit ratio increases during prime time when it is needed most. This is consistent with the observations in Section 10.3 that the share of requests for the most popular programs increases during prime time.

We have here looked at the cache friendliness of the TV-on-demand workload in terms of cacheability and cache hit ratios for the basic LRU and LFU replacement policies. In Section 10.7 on future work we discuss how our observations about access patterns and program popularity can potentially be used to design a more informed caching strategy.

## 10.6 Related Work

There are several studies of viewing behaviour in IPTV systems where traditional scheduled TV is distributed over IP networks. Cha et al. [1] study viewing behaviour including channel popularity and channel switching in an operational IPTV network. Qiu et al. model TV channel popularity [3] and user activities [2] in a large IPTV system. Our work is different in that we look at TV-on-Demand where the viewers choose programs to watch outside of the TV schedule. In this sense our work is closer to studies of traditional VoD systems.

Yu et al. [4] present a large measurement study of the chinese PowerInfo Video-on-Demand system. This work is similar to ours in that they investigate

many aspects of user behaviour and content access patterns. The PowerInfo system is a traditional VoD system. The videos in the library are old TV shows and movies and there are usually only a few new movies introduced to the system per day. This is different from the TV-on-Demand system that we study where there is a large inflow of new programs from the TV-schedule, time-shifted viewing, and programs with a very short life-span. Our work is also different in other aspects in that we investigate how the access pattern depend on genre, we study cacheability and use trace-based simulation to investigate what impact the access patterns have on caching.

There are many other interesting studies of VoD systems and video popularity. Griwodz et al. [10] model long-term popularity of videos on the time scale of days based on VHS rental statistics. Lou et al. [11] give examples of the popularity evolution of video files from a Chinese television station. Tang et al. [12] analyse and model many aspects of media server access. Avramova et al. [13] model the popularity evolution of TV-on-demand and video traces. Dan and Carlsson [7] measure and analyse BitTorrent content popularity. Guo et al. [8] study the probability distributions of Internet media workloads and analyse caching using a mathematical model. Yin et al. [14] study live VoD workloads from the 2008 Beijing Olympics. There are also many studies of Youtube and user generated videos [15, 5, 16, 17].

Gopalakrishnan et al. [18] study user behaviour in a large IPTV system. This is similar to our work but their focus is on modeling the interactive user behaviour in an IPTV environment, including how users fast-forward, pause and rewind to control their viewing.

In this paper we also investigate cacheability and we look at the potential for caching in a TV-on-Demand system. Caching has been widely studied for web content and video [6, 19, 20, 21]. More recently, Ager et al. [9] studied the cacheability for HTTP- and P2P-based applications. There are also several studies of caching strategies in IPTV on-demand systems [22, 23, 24, 25, 26], but these studies use analytical models and simulations whereas we present a trace-based study from a real TV-on-Demand system.

## 10.7 Future Work

In this paper we have studied many aspects of the access patterns in a TV-on-Demand system. We have looked at the cache friendliness of the workload in terms of cacheability and hit ratios for basic replacement policies. For future work we hope that our observations can be used as a basis for developing better

caching strategies for TV-on-Demand systems.

When studying the cache friendliness of the request stream in Section 10.5 we used the basic LRU and LFU cache replacement policies. With these the last requested program is always cached and the choice of what to evict from the cache is between the least recently and the least frequently requested program. A more advanced system could use more knowledge about access patterns and program popularity to decide what program to put in the cache and what program to evict.

One such strategy could be to keep track of all programs in the system, also those that are not currently in the cache. One could monitor the popularity by counting requests, let the programs age over time and for each program keep a value that describes the probability that it will be requested. There are several observations in this paper that can be useful for such an informed caching strategy:

**Give preference to new programs**
With time-shifted TV ongoing scheduled programs immediately get a lot of requests. Some programs, like TV-news, also have a very short life-span. The value of a program should not have to be built up by requests over a long time.

**Categorize programs by genre to predict change in popularity over time**
We saw in Section 10.4 that the access pattern very much depends on the type of program. A news program that is top-ranked the first evening age quickly and have a very low probability for being requested the next evening. A rental movie however is popular for months and increase in rank during weekends. By categorizing programs by genre the probability for future requests can be predicted. The categorization of programs can also be more detailed. The request patterns for different episodes of the same show are often very similar as we saw in Figure 10.9, Section 10.4.2. For a new episode of a show it is a reasonable assumption that the popularity of the program will change over time in a way similar to that of the previous episodes.

**Focus on prime time**
The value of a program should reflect the probability that it will be requested during prime time. There are large peaks in demand in the evenings and at the weekends that need to be handled. If caching is used to limit the maximum link load then it is essential to have the right programs in the cache on Friday and Saturday evenings. There are program like cartoons that are top-ranked in the mornings and early evenings that probably should not be in the cache.

The observations and the predictions outlined above can be used to optimise the caching performance. However, the basic monitoring of request fre-

quency is still needed as a basis, and to handle unexpected changes and sudden peaks in program demand for instance due to large news events.

## 10.8 Conclusions

We have analysed the access patterns in a large TV-on-Demand system and studied the potential for caching.

Our contribution in this paper is three-fold. As a first-order result, we provide reconfirmation of known observations with an independent dataset. We demonstrate that there is a small set of programs that account for a large part of the requests. The program popularity conforms with the Pareto principle, or 80-20 rule. The demand follows a diurnal and weekly pattern, and there are large peaks in demand on Friday and Saturday evenings that need to be handled.

Second, we provide systematic evidence of TV-on-Demand access pattern characteristics that are intuitive yet unconfirmed in the literature. We show that news programs have a very short lifespan and are often only requested for a few hours, children's programs are top ranked in the mornings and early evenings, and movie rentals are concentrated over weekends.

Finally, we also provide novel insights into access patterns that have not been reported previously to the best of our knowledge. We show how the popularity of TV-on-Demand programs changes over time. We see that the access pattern in a TV-on-Demand system very much depend on what type of content it offers. Furthermore, we find that the share of requests for the top most popular programs grows during prime time, and the change rate among them decreases. The cacheability is very high and the cache hit ratio increases during prime time when it is needed most.

We believe that these observations and findings can guide the design of future systems for TV-on-Demand infrastructures.

## 10.9 Acknowledgments

This work has been performed within the SICS Center for Networked Systems funded by VINNOVA, KKS, SSF, ABB, Ericsson, Saab SDS, TeliaSonera, T2Data, Vendolocus and Peerialism.

# Bibliography


[1] Meeyoung Cha, Pablo Rodriguez, Jon Crowcroft, Sue Moon, and Xavier Amatriain. Watching Television Over an IP Network. In *Proceedings of Internet Measurement Conference(IMC)*, pages 71–84, Greece, October 2008.

[2] Tongqing Qiu, Zihui Ge, Seungjoon Lee, Jia Wang, Jun Xu, and Qi Zhao. Modeling User Activities in a Large IPTV System. In *Proceedings of Internet Measurement Conference*, pages 430–441, Chicago, USA, November 2009.

[3] Tongqing Qiu, Zihui Ge, Seungjoon Lee, Jia Wang, Qi Zhao, and Jun Xu. Modeling Channel Popularity Dynamics in a Large IPTV System. In *Proceedings of SIGMETRICS*, pages 275–286, Seattle, USA, June 2009.

[4] H. Yu, D. Zheng, B. Zhao, and W. Zheng. Understanding User Behavior in Large-Scale Video-on-Demand Systems. In *Proceedings of EuroSys2006*, pages 333–344, Leuven, Belgium, 2006.

[5] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon. Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems. *IEEE Transactions on Networking*, 17:1357–1370, October 2009.

[6] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of IEEE INFOCOM*, 1999.

[7] György Dan and Niklas Carlsson. Power-law Revisited: A Large Scale Measurement Study of P2P Content Popularity. In *Proceedings of the 9th international conference on Peer-to-peer systems (IPTPS'10)*, San Jose, USA, April 2010.

[8] Lei Guo, Enhua Tan, Songqing Chen, Zhen Xiao, and Xiaodong Zhang. The stretched exponential distribution of internet media access patterns. In *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing (PODC'08)*, New York, USA, 2008.

[9] Bernhard Ager, Fabian Schneider, Juhoon Kim, and Anja Feldmann. Revisiting Cacheability in Times of User Generated Content. In *Proceedings of 13th IEEE Global Internet Symposium*, San Diego, CA, USA, March 2010.

[10] C. Griwodz, Michael Bär, and L. Wolf. Long-term Movie Popularity Models in Video-on-Demand Systems or The Life of an on-Demand Movie. In *Proceedings of the fifth ACM international conference on Multimedia*, pages 349 – 357, Seattle, USA, 1997.

[11] J. Lou, Y. Tang, M. Zhang, and S. Yang. Characterizing User Behavior Model to Evaluate Hard Cache in Peer-to-Peer Based Video-on-demand Service. In *Proceedings of MMM'07*, pages 125–134, 2007.

[12] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat. Modeling and Generating Realistic Streaming Media Server Workloads. *Computer Networks*, 51:336–356, 2007.

[13] Z. Avramova, S. Wittevrongel, H. Bruneel, and D. Vleeschauwer. Analysis and Modeling of Video Popularity Evolution in Various Online Video Content Systems: Power Law versus Exponential Decay. In *Proceedings of International Conference on Evolving Internet*, 2009.

[14] H. Yin, X. Liu, F. Qiu, N. Xia, C. Lin, H. Zhang, V. Sekar, and G. Min. Inside the Bird's Nest: Measurements of Large-Scale Live VoD from the 2008 Olympics. In *Proceedings of Internet Measurement Conference (IMC'09)*, USA, 2009.

[15] Y. Borghol, S. Mitra, S. Ardon, N. Carlsson, D. Eager, and A. Mahanti. Characterizing and Modeling Popularity of User-generated Videos. In *Proceedings of IFIP International Symposium on Computer Performance, Modeling, Measurements and Evaluation (PERFORMANCE)*, 2011.

[16] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. YouTube Traffic Characterization: A view From the Edge. In *Proceedings of ACM SIGCOMM Internet Measurement Conference*, San Diego, USA, October 2007.

[17] X. Kang, H. Zhang, G. Jiang, H. Chen, X. Meng, and K. Yoshihira. Measurement, Modeling, and Analysis of Internet Video Sharing Site Workload: A Case Study. In *Proceedings of IEEE Conference on Web Services*, 2008.

[18] V. Gopalakrishnan, R. Jana, K. Ramakrishnan, D. Swayne, and V. Vaishampayan. Understanding Couch Potatoes: Measurement and Modeling of Interactive Usage of IPTV at large scale. In *Proceedings of Internet Measurement Conference (IMC'11)*, 2011.

[19] Jiangchuan Liu and Jianliang Xu. Proxy Caching for Media Streaming Over the Internet. *IEEE Communications Magazine*, 42:88–94, 2004.

[20] Jia Wang. A Survey of Web Caching Schemes for the Internet. *ACM SIGCOMM Computer Communication Review*, 29:36–46, 1999.

[21] Alec Wolman, Geoffrey M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On the scale and performance of cooperative Web proxy caching. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP '99)*, 1999.

[22] Henrik Abrahamsson and Mats Björkman. Simulation of IPTV caching strategies. In *Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS10)*, Ottawa, Canada, July 2010.

[23] Sem Borst, Varun Gupta, and Anwar Walid. Distributed caching algorithms for content distribution networks. In *Proceedings of INFOCOM'10*, San Diego, USA, 2010.

[24] D. De Vleeschauwer, Z. Avramova, S. Wittevrongel, and H. Brueel. Transport Capacity for a Catch-up Television Service. In *Proceedings of EuroITV'09*, pages 161–170, Leuven, Belgium, June 2009.

[25] Danny De Vleeschauwer and Koenraad Laevens. Performance of caching algorithms for IPTV on-demand services. *IEEE Transactions on broadcasting*, 55:491 – 501, 2009.

[26] T. Wauters, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, and E. Six. Co-operative Proxy Caching Algorithms for Time-Shifted IPTV Services. In *Proceedings of 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pages 379–386, Dubrovnik, Croatia, September 2006.

# Swedish Institute of Computer Science

SICS Dissertation Series

01: Bogumil Hausman, Pruning and Speculative Work in OR-Parallel PROLOG, 1990.

02: Mats Carlsson, Design and Implementation of an OR-Parallel Prolog Engine, 1990.

03: Nabiel A. Elshiewy, Robust Coordinated Reactive Computing in SANDRA, 1990.

04: Dan Sahlin, An Automatic Partial Evaluator for Full Prolog, 1991.

05: Hans A. Hansson, Time and Probability in Formal Design of Distributed Systems, 1991.

06: Peter Sjödin, From LOTOS Specifications to Distributed Implementations, 1991.

07: Roland Karlsson, A High Performance OR-parallel Prolog System, 1992.

08: Erik Hagersten, Toward Scalable Cache Only Memory Architectures, 1992.

09: Lars-Henrik Eriksson, Finitary Partial Inductive Definitions and General Logic, 1993.

10: Mats Björkman, Architectures for High Performance Communication, 1993.

11: Stephen Pink, Measurement, Implementation, and Optimization of Internet Protocols, 1993.

12: Martin Aronsson, GCLA. The Design, Use, and Implementation of a Program Development System, 1993.

13: Christer Samuelsson, Fast Natural-Language Parsing Using Explanation-Based Learning, 1994.

14: Sverker Jansson, AKL - - A Multiparadigm Programming Language, 1994.

15: Fredrik Orava, On the Formal Analysis of Telecommunication Protocols, 1994.

16: Torbjörn Keisu, Tree Constraints, 1994.

17: Olof Hagsand, Computer and Communication Support for Interactive Distributed Applications, 1995.

18: Björn Carlsson, Compiling and Executing Finite Domain Constraints, 1995.

19: Per Kreuger, Computational Issues in Calculi of Partial Inductive Definitions, 1995.

20: Annika Waern, Recognising Human Plans: Issues for Plan Recognition in Human-Computer Interaction, 1996.

21: Björn Gambäck, Processing Swedish Sentences: A Unification-Based Grammar and Some Applications, 1997.

22: Klas Orsvärn, Knowledge Modelling with Libraries of Task Decomposition Methods, 1996.

23: Kia Höök, A Glass Box Approach to Adaptive Hypermedia, 1996.

24: Bengt Ahlgren, Improving Computer Communication Performance by Reducing Memory Bandwidth Consumption, 1997.

25: Johan Montelius, Exploiting Fine-grain Parallelism in Concurrent Constraint Languages, 1997.

26: Jussi Karlgren, Stylistic experiments in information retrieval, 2000.

27: Ashley Saulsbury, Attacking Latency Bottlenecks in Distributed Shared Memory Systems, 1999.

28: Kristian Simsarian, Toward Human Robot Collaboration, 2000.

29: Lars-Åke Fredlund, A Framework for Reasoning about Erlang Code, 2001.

30: Thiemo Voigt, Architectures for Service Differentiation in Overloaded Internet Servers, 2002.

31: Fredrik Espinoza, Individual Service Provisioning, 2003.

32: Lars Rasmusson, Network capacity sharing with QoS as a financial derivative pricing problem: algorithms and network design, 2002.

33: Martin Svensson, Defining, Designing and Evaluating Social Navigation, 2003.

34: Joe Armstrong, Making reliable distributed systems in the presence of software errors, 2003.

35: Emmanuel Frécon, DIVE on the Internet, 2004.

36: Rickard Cöster, Algorithms and Representations for Personalised Information Access, 2005.

37: Per Brand, The Design Philosophy of Distributed Programming Systems: the Mozart Experience, 2005.

38: Sameh El-Ansary, Designs and Analyses in Structured Peer-to-Peer Systems, 2005.

39: Erik Klintskog, Generic Distribution Support for Programming Systems, 2005.

40: Markus Bylund, A Design Rationale for Pervasive Computing User Experience, Contextual Change, and Technical Requirements, 2005.

41: Åsa Rudström, Co-Construction of hybrid spaces, 2005.

42: Babak Sadighi Firozabadi, Decentralised Privilege Management for Access Control, 2005.

43: Marie Sjölinder, Age-related Cognitive Decline and Navigation in Electronic Environments, 2006.

44: Magnus Sahlgren, The Word-Space Model: Using Distributional Analysis to Represent Syntagmatic and Paradigmatic Relations between Words in High-dimensional Vector Spaces, 2006.

45: Ali Ghodsi, Distributed k-ary System: Algorithms for Distributed Hash Tables, 2006.

46: Stina Nylander, Design and Implementation of Multi-Device Services, 2007.

47: Adam Dunkels, Programming Memory-Constrained Networked Embedded Systems, 2007.

48: Jarmo Laaksolahti, Plot, Spectacle, and Experience: Contributions to the Design and Evaluation of Interactive Storytelling, 2008.

49: Daniel Gillblad, On Practical Machine Learning and Data Analysis, 2008.

50: Fredrik Olsson, Bootstrapping Named Entity Annotation by Means of Active Machine Learning: a Method for Creating Corpora, 2008.

51: Ian Marsh, Quality Aspects of Internet Telephony, 2009.

52: Markus Bohlin, A Study of Combinatorial Optimization Problems in Industrial Computer Systems, 2009.

53: Petra Sundström, Designing Affective Loop Experiences, 2010.

54: Anders Gunnar, Aspects of Proactive Traffic Engineering in IP Networks, 2011.

55: Preben Hansen, Task-based Information Seeking and Retrieval in the Patent Domain: Process and Relationships, 2011.

56: Fredrik Österlind, Improving Low-Power Wireless Protocols with Timing-Accurate Simulation, 2011.

57: Ahmad Al-Shishtawy, Self-Management for Large-Scale Distributed Systems, 2012.

58: Henrik Abrahamsson, Network Overload Avoidance by Traffic Engineering and Content Caching, 2012.